Cardiac Electrophysiology

A Project for the Course on Numerical Methods for Partial Differential Equations

Sabrina Cesaroni   Melanie Tonarelli   Tommaso Trabacchin

0 Introduction

Cardiac electrophysiology is a crucial area of study that investigates the electrical activities of the heart, which are fundamental to its function. The electrical behavior of cardiac tissue is often modeled mathematically to gain insights into the mechanisms underlying normal and abnormal heart rhythms. The standard mathematical model used in this field is the bidomain model, which conceptualizes the cardiac tissue as two overlapping spaces: the intracellular and extracellular compartments. These compartments interact through the cell membrane, leading to the generation and propagation of electrical signals.

However, due to the complexity of solving the full bidomain model, a simplified approach known as the monodomain model is often employed. This model assumes equal anisotropic conductivities in the intra- and extracellular spaces, allowing the problem to be reduced to solving a single reaction-diffusion equation for the transmembrane potential—the voltage difference across the cell membrane. The monodomain equation is complemented by a model that describes the ionic currents passing through the ion channels, which are responsible for the generation of the action potential, as described in [4].

In this project, we focus on the parallel implementation of the macroscopic monodomain model discussed in [3] coupled with the Bueno-Orovio ionic model described in [2]. The parallelization of these models is essential for efficiently simulating large-scale cardiac tissues, which can provide deeper insights into the dynamics of cardiac electrophysiology and contribute to the development of better diagnostic and therapeutic tools.

1 Problem model definition

Let $\mathbf{w} = (v, w, s)$ be the vector of three gating variables, u the adimensional transmembrane potential and J_{ion} the net ionic current, the system of ODEs describing the **Bueno-Orovio model** is the following:

$$\begin{cases} \dfrac{\mathrm{d}v}{\mathrm{d}t} = \dfrac{(1 - H(u-\theta_v))(v_\infty - v)}{\tau_v^-} - \dfrac{H(u-\theta_v)v}{\tau_v^+} & t > 0 \\ \dfrac{\mathrm{d}w}{\mathrm{d}t} = \dfrac{(1 - H(u-\theta_w))(w_\infty - w)}{\tau_w^-} - \dfrac{H(u-\theta_w)w}{\tau_w^+} & t > 0 \\ \dfrac{\mathrm{d}s}{\mathrm{d}t} = \dfrac{1}{\tau_s}\left(\dfrac{1+\tanh(k_s(u-u_s))}{2} - s\right) & t > 0 \\ v(0) = 1 \\ w(0) = 1 \\ s(0) = 0 \end{cases} \tag{1}$$

where $H(x)$ is the standard Heaviside function and all the time costants are functions of u as described in [2]:

$$\begin{aligned}
\tau_v^- &= (1 - H(u - \theta_v^-))\tau_{v1}^- + H(u - \theta_v^-)\tau_{v2}^- \\
\tau_w^- &= \tau_v^- + (\tau_w^- - \tau_v^-)(1 + \tanh\big(k_w^-(u - u_w^-)\big))/2 \\
\tau_{so} &= \tau_{so1} + (\tau_{so2} - \tau_{so1})(1 + \tanh(k_{so}(u - u_{so})))/2 \\
\tau_s &= (1 - H(u - \theta_w))\tau_{s1} + H(u - \theta_w)\tau_{s2} \\
\tau_o &= (1 - H(u - \theta_o))\tau_{o1} + H(u - \theta_o)\tau_{o2}
\end{aligned}$$

and the infinity values are defined as:

$$\begin{aligned}
v_\infty &= \begin{cases} 1, u < \tau_v^- \\ 0, u \geq \tau_v^- \end{cases} \\
w_\infty &= (1 - H(u - \theta_o))(1 - u/\tau_{w\infty}) + H(u - \theta_o)w_\infty^*
\end{aligned}$$

The above model (1) can be written in a more compact way as

$$\frac{\partial \mathbf{w}}{\partial t} = \mathbf{F}_{\mathrm{BO}(u,\mathbf{w})}$$

Moreover, in this model the ionic current is made up of three components $J_{ion} = J_{fi} + J_{so} + J_{si}$, such that:

$$\begin{aligned}
J_{fi} &= -vH(u - \theta_v)(u - \theta_v)(u_u - u)/\tau_{fi} \\
J_{so} &= (u - u_0)(1 - H(u - \theta_w))/\tau_o + H(u - \theta_w)/\tau_{so} \\
J_{si} &= -H(u - \theta_w)ws/\tau_{si}
\end{aligned} \tag{2}$$

The **monodomain model** equations described in [3], coupled with the ionic model and the initial conditions, are:

$$\begin{cases}
C_m\chi\dfrac{\partial V}{\partial t} - \boldsymbol{\nabla}\cdot(\boldsymbol{\sigma}\boldsymbol{\nabla}V) - C_m\chi J_{ion}(V, \mathbf{w}) = I_{app} & \text{in } \Omega \\
\boldsymbol{\sigma}\boldsymbol{\nabla}V\mathbf{n} = 0 & \text{on } \partial\Omega \\
V(t = 0) = V_0 & \text{in } \Omega \\
\dfrac{\mathrm{d}\mathbf{w}}{\mathrm{d}t} = \mathbf{F}_{\mathrm{BO}}(V) & \text{in } \Omega \\
\mathbf{w}(t = 0) = \mathbf{w}_0 & \text{in } \Omega
\end{cases} \tag{3}$$

where Ω is the computational domain, $\boldsymbol{\sigma}$ is the tissue conductivity tensor, I_{app} is the applied external stimulus current, C_m is the membrane capacitance and χ is the cellular surface-to-volume ratio.

Note that V is expressed in milliVolt and u is dimensionless, and the two quantities are related according to the following formula:

$$V = 85.7u - 84$$

1.1 Weak formulation of the monodomain model

Let $W = H^1(\Omega)$ and $v \in W$. Multiplying the reaction-diffusion equation in (3) by v, a *test function*, and then integrating by parts and considering the homogeneous Neumann boundary conditions, we obtain:

$$\int_\Omega C_m\chi\frac{\partial V}{\partial t}v\,\mathrm{d}\mathbf{x} + \int_\Omega \boldsymbol{\sigma}\boldsymbol{\nabla}V\cdot\boldsymbol{\nabla}v\,\mathrm{d}\mathbf{x} - \int_\Omega C_m\chi J_{ion}(V, \boldsymbol{w})v\,\mathrm{d}\mathbf{x} = \int_\Omega I_{app}v\,\mathrm{d}\mathbf{x}$$

We introduce the notation

$$a : V \times V \to \mathbb{R} \ \text{ s.t. } a(V, v) = \int_\Omega \boldsymbol{\sigma}\boldsymbol{\nabla}V\cdot\boldsymbol{\nabla}v \ \mathrm{d}\mathbf{x}$$

$$F : V \to \mathbb{R} \text{ s.t. } F(v) = \int_\Omega I_{app} v \, \mathrm{d}\mathbf{x}$$

Then, the weak formulation reads:

$$\forall t \in (0,T) \text{ find } V(t) \in W \text{ such that } V(0) = V_0 \text{ and}$$
$$\int_\Omega C_m \chi \frac{\partial V}{\partial t} v \, \mathrm{d}\mathbf{x} + a(V,v) - \int_\Omega C_m \chi J_{ion}(V, \boldsymbol{w}) v \, \mathrm{d}\mathbf{x} = F(v) \quad \forall v \in W$$

1.2 Space discretisation

Let us introduce a mesh over the domain Ω and $W_h = X_h^r(\Omega) \cap W$, a finite dimensional subspace of W such that $N_h = \dim(W_h)$, where $X_h^r(\Omega)$ is the space of piecewise polynomials of degree r over the mesh elements. The semi-discrete formulation reads:

$$\forall t \in (0,T) \text{ find } V_h(t) \in W_h \text{ such that} V_h(0) = V_{0,h} \text{ and}$$
$$\int_\Omega C_m \chi \frac{\partial V_h}{\partial t} v_h \, \mathrm{d}\mathbf{x} + a(V_h, v_h) - \int_\Omega C_m \chi J_{ion}^h(V_h, \boldsymbol{w}_h) v_h \, \mathrm{d}\mathbf{x} = F(v_h) \quad \forall v_h \in W_h,$$
$$\text{with } V_{0,h} \in V_h \text{ an approximation of } V_0.$$

Introducing the basis functions $\{\varphi_i\}_{i=1}^{N_h}$ of the space W_h, we can write

$$V_h(\mathbf{x}, t) = \sum_{j=0}^{N_h} u_j^h(t) \varphi_j(\mathbf{x})$$

and exploiting the linearity of the involved bilinear forms and operators, the semidiscrete formulation can be rewritten as the following system of ODEs:

$$\begin{cases} C_m \chi M \dfrac{\mathrm{d}\mathbf{V}(t)}{\mathrm{d}t} + K\mathbf{V}(t) + C_m \chi \mathbf{J}_{ion}^h = \mathbf{I}_{app} \\ \mathbf{V}(0) = \mathbf{V}_0 \end{cases} \tag{4}$$

where $\mathbf{V}_0$ is the vector of control variables for $V_{0,h} \in W_h$ and

$$\begin{aligned}
&\mathbf{V}(t) = (u_1^h(t), \ldots, u_{N_h}^h(t))^T \\
&M \in \mathbb{R}^{N_h \times N_h} : \quad (M)_{ij} = (\varphi_j, \varphi_i) \\
&K \in \mathbb{R}^{N_h \times N_h} : \quad (K)_{ij} = a(\varphi_j, \varphi_i) \\
&\mathbf{J}_{ion}^h \in \mathbb{R}^{N_h} : \quad (\mathbf{J}_{ion}^h)_i = (J_{ion}(V_h, \mathbf{w}), \varphi_i) \\
&\mathbf{I}_{app} \in \mathbb{R}^{N_h} : \quad (\mathbf{I}_{app})_i = F(\varphi_i)
\end{aligned}$$

Note that the spacial approximation of the gating variables $\mathbf{w}(t) \in W$, needed for the computation of $\mathbf{J}_{ion}^h$, can be obtained with different approaches:

1. In the first approach, called nodal interpolation, one looks for an approximated gating variable $\tilde{\mathbf{w}} \in W_h$ so that we can write

   $$\tilde{\mathbf{w}}(t) = \sum_{i=1}^{N_h} \varphi_i(\mathbf{x}) \tilde{\mathbf{w}}_i(t)$$

   Since the $\tilde{\mathbf{w}}$ affects the monodomain equation through the J_{ion} term which requires to be evaluated at the quadrature nodes $\{\mathbf{x}_q\}_{q=1}^{n_{qn}}$ of the mesh for the approximation of the integral, then there are two possibilities to do so:

- the first method is called **state variables interpolation** (SVI) for which both the potential variable V_h and the gating variables $\mathbf{w}_h$ are interpolated at the quadrature nodes of the mesh. Therefore, the ionic current at each quadrature node can be computed as shown in equation 2:

$$J_{ion}^{SVI}(\mathbf{x}_q) = J_{ion}\left(\sum_{i=1}^{N_h} \varphi_i(\mathbf{x}_q) u_i^h(t), \sum_{i=1}^{N_h} \varphi_i(\mathbf{x}_q)\tilde{\mathbf{w}}_i(t)\right)$$

- Another method is the **ionic current interpolation** (ICI) for which the ionic current is first evaluated at the control variables, computed at each degree of freedom of the mesh and then interpolated on each quadrature node. Therefore,

$$J_{ion}^{ICI}(\mathbf{x}_q) = \sum_{i=1}^{N_h} \varphi_i(\mathbf{x}_q) J_{ion}\big(u_i^h(t), \tilde{\mathbf{w}}_i(t)\big)$$

2. The second approach is called **Gauss integration** (GI) which consists of defininine each gating variable and then solving the system 1 at each quadrature node. In particular, by denoting a vector $\overline{\mathbf{w}}(t) = (\overline{\mathbf{w}}_1(t), \ldots, \overline{\mathbf{w}}_{nq}(t)) \in \mathbb{R}^{n_{nq}}$ of n_{nq} control variables defined correspondingly to the quadrature nodes $\{\mathbf{x}_q\}_{q=1}^{n_{qn}}$, the ionic current can be computed simirarly to the ICI approach:

$$J_{ion}^{GI}(\mathbf{x}_q) = \sum_{i=1}^{N_h} \varphi_i(\mathbf{x}_q) J_{ion}\big(u_i^h(t), \overline{\mathbf{w}}_q(t)\big)$$

We remark that the approach chosen for the spatial approximation of the gating variable $\mathbf{w}(t)$ may have significant consequences on the accuracy of the solution of the full coupled monodomain problem, as discussed in [4].

1.3 Time discretisation

The time discretization of the monodomain equation is not trivial, as the J_{ion} term is non-linear with respect to u. Therefore, a fully implicit scheme would involve solving a computationally-expensive non-linear problem at each timestep. On the other hand, a fully explicit approach could potentially introduce instabilities.

For these reasons, a semi-implicit method has been adopted. In particular, the diffusion term and forcing term are treated using the θ-method, while the ionic current term is treated fully explicitly.

Let us introduce a partition of the time interval $[0, T]$ into N_t time sub-intervals $(t^k, t^{k+1}]$ of width Δt, with $n = 0, ..., N_t - 1$ and such that $t^0 = 0$, $t^{N_T} = T$ and $t^k = k\Delta t$. At each time step k, firstly the system of ODEs 1 is solved and then the ionic current is computed (according to the chosen coupling). By denoting $\mathbf{V}(t^k)$ as $\mathbf{V}^K$ and $\theta \in [0, 1]$ the parameter of the θ-method, the fully-discrete formulation reads as:

$$\begin{cases} C_m \chi M \dfrac{\mathbf{V}^{k+1} - \mathbf{V}^k}{\Delta t} + \theta K \mathbf{V}^{k+1} + (1-\theta) K \mathbf{V}^k = \theta \mathbf{I}_{app}^{k+1} + (1-\theta)\mathbf{I}_{app}^k - C_m \chi \mathbf{J}_{ion} & \forall k = 0, \ldots, N_T - 1 \\ \mathbf{V}^0 = \mathbf{V}_0 \end{cases}$$

2 Code Implementation

The project is composed of three primary modules:

- The **Ionic Model**, which computes the gating variables and ionic currents at each time step.
- The **Coupler**, which integrates the Ionic Model with the Monodomain Solver.

- The **Monodomain Solver**, which solves the monodomain equation using the finite element method.

Additionally, there is a class named `Solver` that orchestrates the interactions among these three main modules.

In the following subsections, each module and the parallel implementation technique will be discussed in greater detail.

2.1 Parallel implementation

In order to take advantage of multiprocessors systems, our implementation exploits the MPI features integrated into the `deal.II` library. In particular, as described in [1], the mesh and the degrees of freedom are distributed. This means that each process stores a share of the cells and degrees of freedom, which consists of exactly those elements that it owns, as well as one layer of ghost elements around the owned ones. Note that ghost elements are read-only. Therefore, in order to ensure that all data are up-to-date, inter-process communication must take place. The exact communication pattern for each module will be discussed below.

2.2 Ionic Model

The ionic model is built around an abstract class, `IonicModel`, which serves as a virtual representation of a generic ionic model. This class provides a method to compute the ionic current given the dimensionless transmembrane voltage u and the state variables. Additionally, it offers a method to compute the derivative of each state variable based on these parameters.

For this project, the Bueno-Orovio (BO) model was selected, thus a subclass implementing this specific model is provided.

Furthermore, the module includes the `ODESolver` class, which is used to solve the ODEs system associated to the ionic model using the θ-method.

Given that the equations 1 in the BO model are linear with respect to the gating variables but not with respect to the dimensionless voltage u, each equation can be expressed as follows by discretising the first derivative using the incremental ratio:

$$\frac{x^{k+1} - x^k}{\Delta t} = \alpha(u^k)x^{k+1} + \beta(u^k) \qquad \forall k = 0, \dots, N_T - 1$$

where x represents a generic state variable, Δt the time step, and $\alpha(u^k)$ and $\beta(u^k)$ are functions of u^k and are referred to as the implicit and explicit terms, respectively.
This formulation allows each state variable to be treated implicitly. The implications of this approach will be discussed in the following section.

This module does not perform any inter-process communication.

2.3 Monodomain Solver

The monodomain solver is the component responsible for solving the monodomain equation, using the Finite Element Method (FEM). The component is implemented through a single class (`FESolver`) using the `deal.II` library together with the sparse linear algebra `Trilinons` library. The class offers several methods:

- `setup` method, used to setup all vectors and matrices. Moreover, it also assembles the M and K matrices, which do not depend on the current time step, and that therefore can be assembled upfront;
- `assemble_rhs` method, that, at each time step, assembles the right-hand side of the linear system;

- `solve_time_step` method, which solves, at each time step, the assembled linear system, applying a suitable preconditioner, whose exact choice will be discussed in section 4.2;
- `output` method, responsible for printing on a file the current solution. Given that this method is rather expensive, it is not called at each time step, but once every 100 time steps.

In this module, each process needs to communicate its owned values of V, so that ghost values of neighbouring processes are up-to-date.

2.4 Couplers

The Couplers component provides an abstract description of the `Coupler` class. In particular, a coupler is responsible for performing the necessary interpolations on the quadrature nodes of the transmembrane voltage and/or the state variables, depending on the specific coupling strategy chosen among the three ones detailed in section 1.2.
In the module, all possible strategies are implemented in three different classes (`SVICoupler`, `ICICoupler`, `GICoupler`).

The exact communication pattern depends on the specific coupler:

- with both `SVICoupler` and `ICICoupler`, processes need to communicate the gating variables to the others one.
- with `GICoupler` no inter-process communication is required.

3 Verification and Accuracy

3.1 Methods

In order to verify the correctness of the proposed implementation, several tests have been conducted according to the benchmark proposed in [3], taking into account all couplers. In particular, the primary metric for the benchmark is the activation time which is, for each point in the mesh, the time needed by the membrane potential V to be raised from its resting value (about -84mV) to a positive value. Specifically, all tests required by the benchmark have been executed with all possible couplers.

The table 1 summarises the details of the conducted benchmark, while Figure 1 shows the schematic drawning of the model.

variable	description
geometric domain	cuboid
dimensions	$20 \times 7 \times 3$ mm
fibre orientation	fibres are aligned in the long, 20mm, axis
discretizations Δx	0.5, 0.2 and 0.1 isotropic
PDE time steps Δt	0.05, 0.01 and 0.005 ms
θ_{ODE}, θ_{FE}	0.5, 0.5
stimulation geometry	$1.5 \times 1.5 \times 1.5$mm cube from a corner
stimulation protocol	2ms at 50 000μAcm^{-3}
surface area to volume ratio χ	140mm^{-1}
membrane capacitance C_m	0.01μFmm^{-2}
conductivities σ_i^l, σ_i^t, σ_e^l and σ_e^t	0.17, 0.019, 0.62 and 0.24 Sm^{-1}
tissue type	epicardium

Table 1: Benchmark description

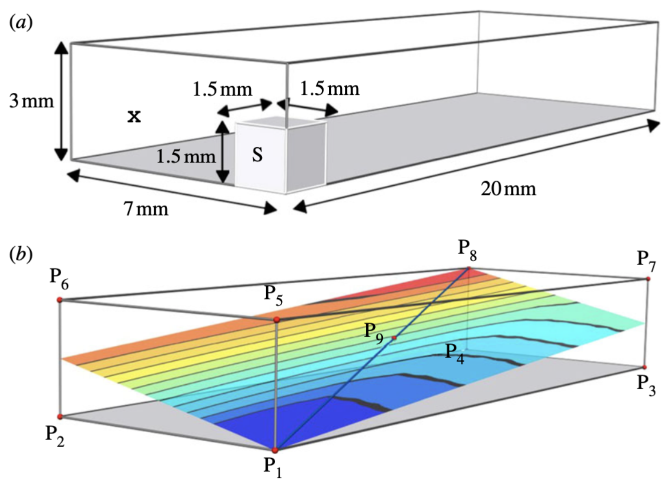


Figure 1: Schematic of the simulation domain and summary of points at which the activation time was evaluated.

3.2 Experimental results

After conducting exhaustive tests, the obtained results are shown in figures 4 and 3.

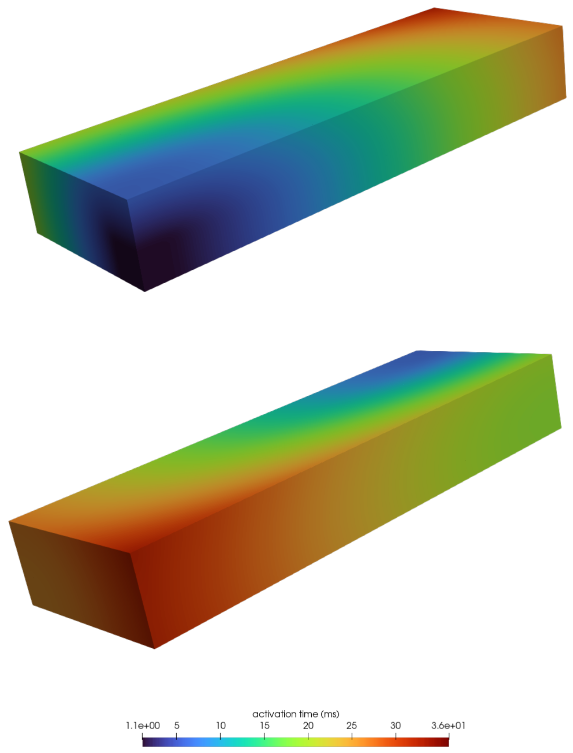
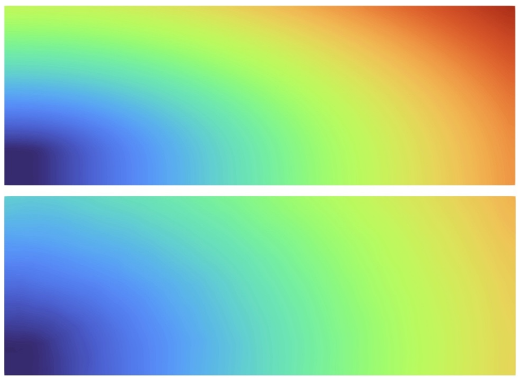
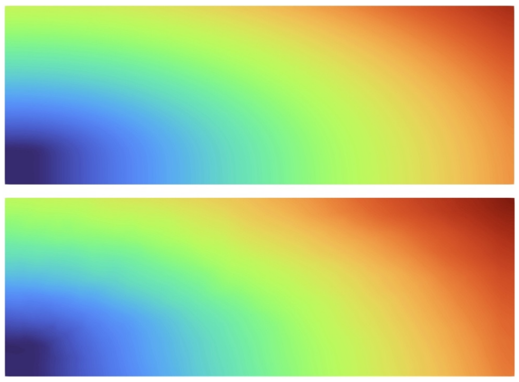


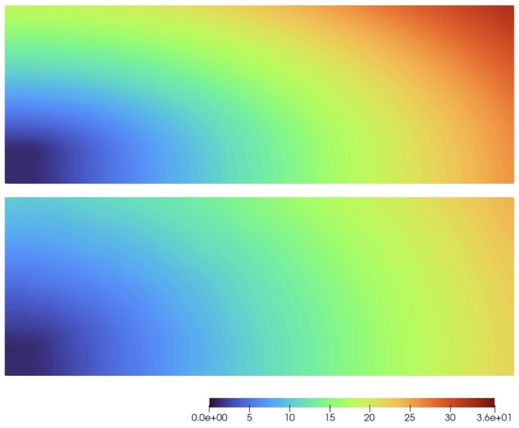
Figure 2: An example of activation times on the cuboid depicted in figure 1. The activation times are represented by the colour map from dark blue (0 ms) to red (36 ms).



(a) GI Coupler



(b) ICI Coupler



(c) SVI Coupler

Figure 3: Activation times on the plane depicted in figure 1 for each coupler. The upper and lower planes correspond to the solutions with Δx = 0.5mm Δt = 0.005ms and Δx = 0.1mm $\Delta t = 0.005$ms, respectively. The activation times are represented by the colour map from dark blue (0 ms) to red (36 ms).

It is worth noticing that in the benchmark proposed in [3], there are two codes (A and H) whose results are closely related to the ones obtained in this project.

In particular, both of them are based on the Finite Element Method (FEM), employ a tetrahedral mesh and do not perform the lumping of the mass matrix. As showned in figure 4 and table 2, it can be observed that the results obtained by our implementation are in good agreement with ones of code A, and quite similar to the ones of code H, which shows a different behaviour only at the coarsest spatial resolution.

Additionally, the difference in accuracy among the various couplers can be analyzed, as illustrated in table 2. According to the obtained data, it is clear that `SVICoupler` and `GICoupler` show a similar behaviour with respect to all mesh refinements, while the use of the `ICICoupler` results in sligthly different activation times. This difference is clearly more relevant for coarser meshes, while it becomes negligible for the finest one. All this is true independently of the chosen time step Δt, as long as it is below 0.05ms. When a larger time step was chosen, the results obtained were not physiologically realistic, indicating the presence of numerical instabilities, due to the fact that a semi-implicit approach is employed. Meanwhile when a time step between 0.05ms and 0.005ms was chosen, the results in terms of activation times were only slightly different with respect to

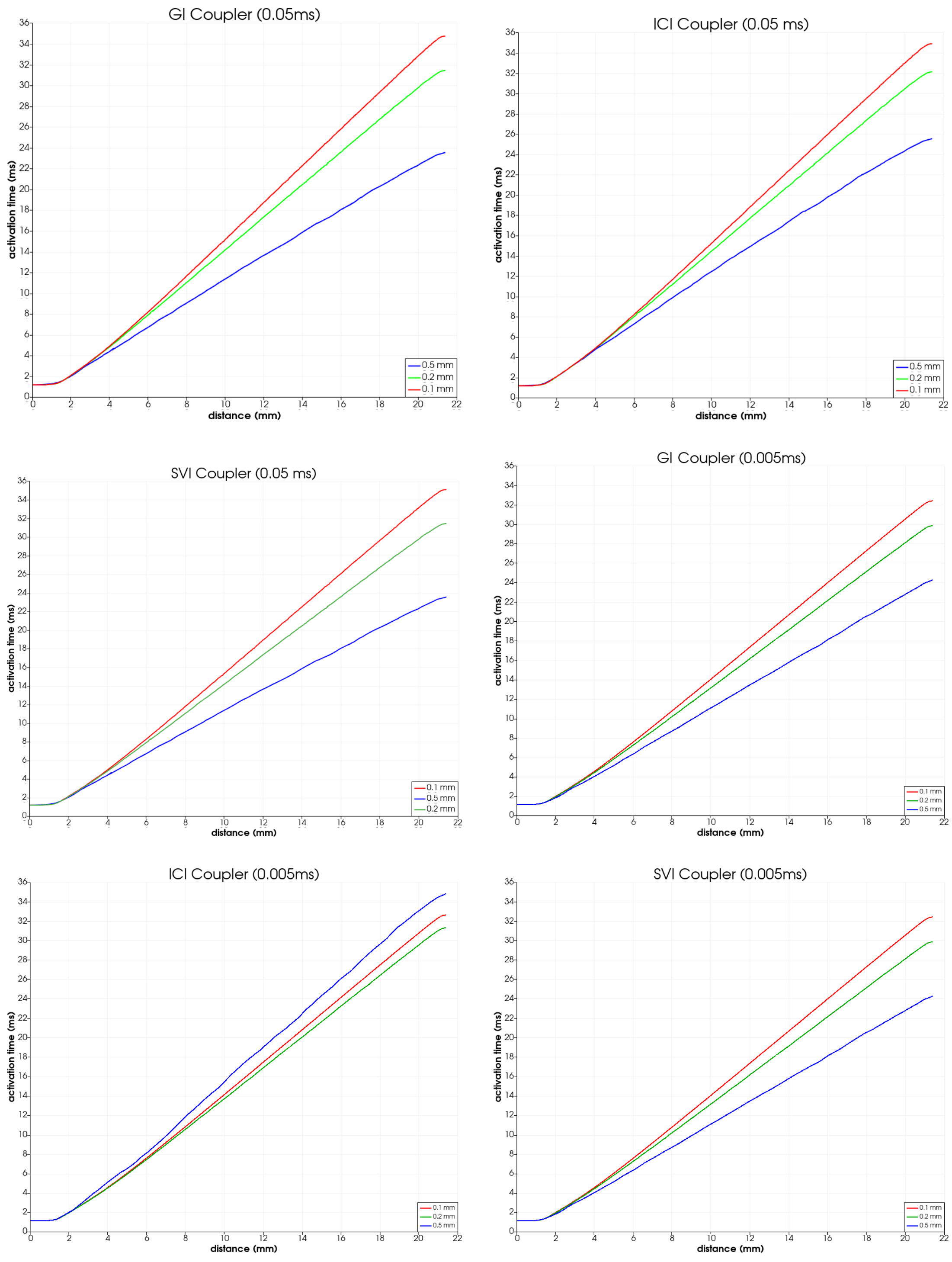


Figure 4: Activation times in ms along the line P1-P8 in figure 1 for solutions with $\Delta t = 0.05$ms and 0.005ms and $\Delta x = 0.1$mm (red line), 0.2mm (green line) and 0.5mm (blue line).

the 0.005ms case. In particular, it is worth noticing that the version with the `ICICoupler` and $\Delta t = 0.005$ms is the one with the most significantly different result; in this case, the activation times along the line P1-P8 is higher when considering the coarsest mesh, which is a behaviour

Coupler	$\Delta x = 0.1$mm	$\Delta x = 0.2$mm	$\Delta x = 0.5$mm
ICICoupler	32.625	31.32	34.81
GICoupler	32.42	29.87	24.265
SVICoupler	32.42	29.87	24.265

Table 2: Activation times in ms at point P_8 of the model in figure 1 when using a $\Delta t = 0.005$ms.

often shown by many implementations (other than A and H) in [3]. The reason for this behaviour remains unclear.

To summarise, it is possible to conclude that our implementation performs as expected by the benchmark with a good accuracy. Thus, our examination will now move to the performance analysis.

4 Perfomance analysis

The proposed implementation has been evaluated with respect to multiple aspects, including mesh refinement, time step size, coupling strategy and used preconditioner. Moreover, strong scalability and execution times are examined.

4.1 Methods

The same model proposed in table 1 is assumed.

The benchmark was conducted on the MeluXina system. A system overview is shown in table 3.

Compute Module	**CPU**	**RAM**
Cluster	2x AMD EPYC Rome 7H12 64c @2.6GHz	512GB

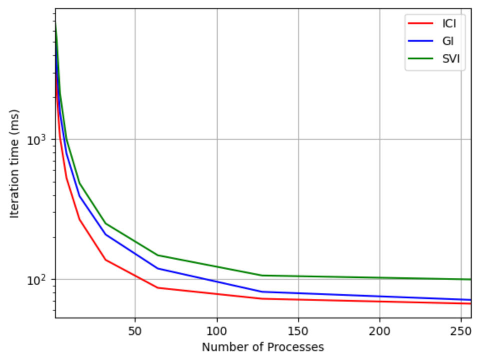
Table 3: MeluXina Compute Modules Overview.

4.2 Experimental results

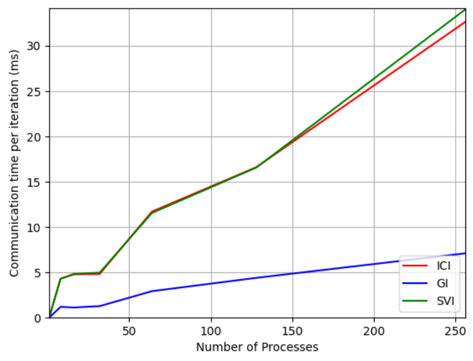
In figure 5, multiple plots are shown in order to analyse different performance aspects of the implementation.

Specifically, the proposed project demonstrates strong scalability, as shown in Figure 5(a), with execution time scaling linearly with the number of processes. The tests were conducted on a single compute node for 32 or fewer processes, while two or more compute nodes were used for more than 32 processes. It's important to note that employing multiple nodes can introduce communication overhead between nodes and processes, which hinders scalability for an higher number of processes as it can be observed in figure 5(a) (the speedup after roughly 128 processes becomes sublinear).

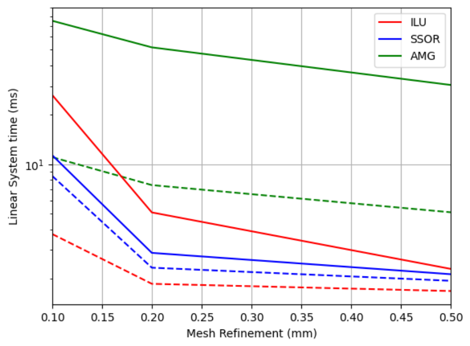
Furthermore, as expected, Figure 5(b) illustrates that communication time increases with the number of processes. It is observed that the `SVICoupler` and `ICICoupler` exhibit similar behaviour in this regard, while the `GICoupler` has significantly lower communication time. This behaviour is expected, as both `SVICoupler` and `ICICoupler` compute the gating variables at the degrees of freedom (DoFs), therefore the computations involving ghost elements require communication between processes as explained in section 2.1, leading to increased execution time. In contrast, the



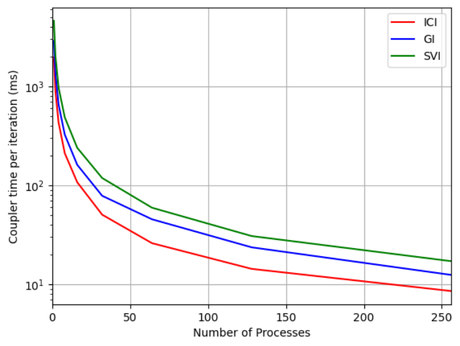
(a) Strong Scalability using a logarithmic scale.



(b) Communication time per time step.



(c) Time to solve the linear system using Conjugate Gradient (CG) method using different preconditioners. Dashed lines represent the time of the CG, meanwhile the thick line represents the sum of the time to construct the preconditioner and the time of the CG.



(d) Time to interpolate according to the different coupler per time step.

Figure 5: Various graphics representing different perfomance analysis of the code implementation using as model the one depicted in 1 with $\Delta x = 0.1$mm and $\Delta t = 0.005$ms.

`GICoupler` calculates the gating variables at the quadrature nodes, with each process managing its own independent subset of nodes. As a result, there is no need for inter-process communication for the gating variables.

To reduce the time required to solve the linear system, derived from the FEM, various preconditioners were tried in conjunction with the Conjugate Gradient method. As shown in Figure 5(c), the SSOR preconditioner emerged as the most efficient option, performing well on both finer and coarser meshes.

The main difference between couplers is the time required to perform the necessary interpolations. As shown in 5(d), the `SVICoupler` is the one that spends the most time on interpolating the gating variables and the potential. Given that the `GICoupler` only needs to interpolate the potential, it can achieve lower interpolation time, with respect to the `SVICoupler`. However, the best interpolation performance is attained by the `ICICoupler`, which only must interpolate the ionic current.

For what concerns the time needed to solve the ODEs system related to the Bueno-Orovio model, it has emerged that it is negligible compared to the total solution time, even for the `GICoupler`.

Coupler Type	**ODE Time / Solve Time**
ICI	0.13%
GI	1.64%
SVI	0.08%

Table 4: Percentage of time spent solving the ODE system versus time spent resolving the time step for each coupler.

5 Couplers comparison

The results obtained in this project allow for a comparison between the three couplers. The `ICICoupler` stands out as the most computationally efficient, offering superior performance compared to the other couplers. However, this efficiency comes at the cost of accuracy, as it produces incorrect results when applied to coarser meshes.

In contrast, both the `SVICoupler` and the `GICoupler` demonstrate good accuracy, with the `GICoupler` outperforming the `SVICoupler` in terms of performance. This suggests that while the `SVICoupler` incurs additional computational overhead due to interpolation and communication, the cost of solving the ODE system at all quadrature nodes, rather than just at the degrees of freedom, is a trade-off that favors the `GICoupler`.

6 Conclusion

In summary, our implementation has demonstrated good accuracy against the benchmark across various mesh refinements, time steps, and couplers. Additionally, the code exhibits strong scalability, making it well-suited for multiprocessor systems. The choice of the most suitable coupler depends on the specific requirements of the simulation: for accuracy, the `GICoupler` is the preferred choice, while for performance, the `ICICoupler` is more appropriate.

A potential extension of this project could involve exploring whether the use of mass lumping techniques, in conjunction with a fully explicit method, could enhance performance without significantly compromising accuracy.

References

[1] The deal.ii library, 2024. `https://dealii.org/current/doxygen/deal.II/`.

[2] Alfonso Bueno-Orovio, Elizabeth Cherry, and Flavio Fenton. Minimal model for human action potentials in tissue. *Journal of theoretical biology*, 2008.

[3] Steven Niederer, Eric Kerfoot, Alan Benson, Miguel Bernabeu, Olivier Bernus, Chris Bradley, Elizabeth Cherry, Richard Clayton, Flavio Fenton, Alan Garny, Elvio Heidenreich, Sander Land, Mary Maleckar, Pras Pathmanathan, Gernot Plank, Jos Rodrguez, Ishani Roy, Frank Sachse, Gunnar Seemann, and Nic Smith. Verification of cardiac tissue electrophysiology simulators using an n-version benchmark. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 2011.

[4] Alessandro S. Patelli, Luca Dedè, Toni Lassila, Andrea Bartezzaghi, and Alfio Quarteroni. Isogeometric approximation of cardiac electrophysiology models on surfaces: An accuracy study with application to the human left atrium. *Computer Methods in Applied Mechanics and Engineering*, 317:248–273, 2017.

[5] Francesco Regazzoni, Matteo Salvador, Pasquale Claudio Africa, Marco Fedele, Luca Dede, and Alfio Quarteroni. A cardiac electromechanical model coupled with a lumped-parameter model for closed-loop blood circulation. *Journal of Computational Physics*, 457:111083, 02 2022.