POLITECNICO
MILANO 1863

Course of Numerical Analysis for Partial Differential Equations

A HIGH-ORDER DISCONTINUOUS GALERKIN METHOD FOR THE BIDOMAIN PROBLEM OF CARDIAC ELECTROPHYSIOLOGY

Authors: Federica Botta, Matteo Calafà

Supervisors: Christian Vergara, Paola Antonietti

A.Y. 2020/2021

Contents

1 Introduction

1.1 Abstract

The aim of the project is to study and implement a suitable numerical scheme for the resolution of the *Bidomain Problem*, a famous system of equations that has been developed in the context of the electrophysiology of human heart.
This work is basically the continuation of a two-years-long study carried out by three past course projects ([3], [1], [7]). In particular, the very goal of this project is to improve the results obtained in [7] (Marta and Perego) for the Bidomain model. In fact, even if a *Discontinuous Galerkin* discretization has been successfully implemented, results are not satisfactory from the point of view of stability and convergence. We think this notice is noteworthy as this work is primarily based on these provided data and codes. Through this article, it will be illustrated how we managed to solve these problems extending, optimizing and correcting these past numerical strategies.

1.2 The physical problem

We intend to present the physical meaning of the Bidomain equations only briefly since it has already been widely shown in the previous project (Marta and Perego). For a more complete explanation, we instead refer to [9].
The mechanical contraction and expansion of human heart has its origin in the *electrical activation* of the cardiac cells. At every heart-beat, myocyties are activated and deactivated following a characteristic electrical cycle (fig 1).

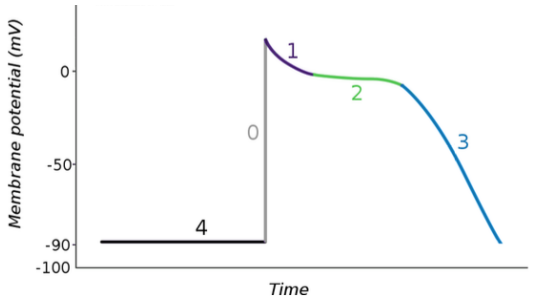

Figure 1: Membrane potential in function of time (one cardiac cycle)

The cell is initially at rest ($-90mV$, step 4). At a certain point, its potential increases rapidly ($\approx 2ms$) and reaches the value of $+20mV$: the cell is activated. Later, a plateau near $0mV$ is observed and then a slow repolarization to the initial potential.
From a microscopical point of view, we could study the dynamics acting in each single cell (as a consequence of the passage of chemical ions through specific channels, e.g. $Ca2+, Na+, K+$). From a macroscopical point of view, instead, one can

observe it as a continuous electrical diffusion over the entire cardiac surface. Even if this consists in a very rapid phenomenon, the study of such propagation could be very interesting in order, for instance, to detect diseases in sick patients.

1.3 Mathematical models

Starting from the circuit in figure 2, applying some general electromagnetism laws and some calculations, the Bidomain model has been formulated (see [9] for more details and/or [5] for the complete passages).

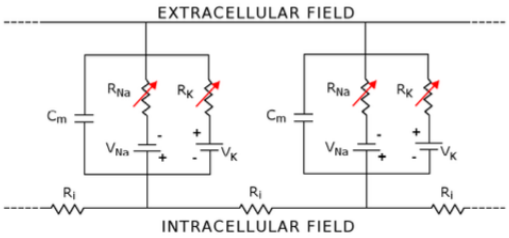

Figure 2: Simplified circuit to model the intracellular and extracellular potentials dynamics

The general formulation is then:

Definition 1 (Bidomain model)**.**

$$\begin{cases} \chi_m C_m \frac{\partial V_m}{\partial t} - \nabla \cdot (\Sigma_i \nabla \phi_i) + \chi_m I_{ion} = I_i^{ext} & in\ \Omega_{mus} \times (0,T] \\ -\chi_m C_m \frac{\partial V_m}{\partial t} - \nabla \cdot (\Sigma_e \nabla \phi_e) - \chi_m I_{ion} = -I_e^{ext} & in\ \Omega_{mus} \times (0,T] \end{cases}$$

where:

- $\boldsymbol{\phi_i, \phi_e}$ are the *Intracellular and Extracelllular Potentials* (unknowns)
- $V_m = \phi_i - \phi_e$ is the *Trans-membrane Potential*
- χ_m, C_m are known constants and Σ_i, Σ_e are known constant tensors
- I_i^{ext}, I_e^{ext} are applied currents
- I_{ion} is the *Ionic Current*
- Ω_{mus} is the cardiac domain (myocardium + endocardium + epicardium)

Actually, this system is not complete since it misses boundary and initial conditions and a suitable model for I_{ion}. Initial conditions and Neumann boundary conditions for ϕ_i and ϕ_e are then imposed. For the definition of I_{ion}, instead, a *reduced ionic model* is chosen, in particular the *FitzHugh-Nagumo model*. Summing up:

Definition 2 (Bidomain + FitzHugh-Nagumo model with Neumann boundary conditions)**.**

$$\begin{cases} \chi_m C_m \frac{\partial V_m}{\partial t} - \nabla \cdot (\Sigma_i \nabla \phi_i) + \chi_m I_{ion}(V_m, w) = I_i^{ext} & \text{in } \Omega_{mus} \times (0,T] \\ -\chi_m C_m \frac{\partial V_m}{\partial t} - \nabla \cdot (\Sigma_e \nabla \phi_e) - \chi_m I_{ion}(V_m, w) = -I_e^{ext} & \text{in } \Omega_{mus} \times (0,T] \\ I_{ion}(V_m, w) = kV_m(V_m - a)(V_m - 1) + w & \text{in } \Omega_{mus} \times (0,T] \\ \frac{\partial w}{\partial t} = \epsilon(V_m - \gamma w) & \text{in } \Omega_{mus} \times (0,T] \\ \Sigma_i \nabla \phi_i \cdot n = b_i & \text{on } \partial\Omega_{mus} \times (0,T] \\ \Sigma_e \nabla \phi_e \cdot n = b_e & \text{on } \partial\Omega_{mus} \times (0,T] \\ \textit{Initial conditions for } \phi_i, \phi_e, w & \text{in } \Omega_{mus} \times \{t = 0\} \end{cases}$$

where:

- $\boldsymbol{w}$ is the *gating variable* (unknown)
- k, a, ϵ, γ are known constants
- b_i, b_e are the boundary conditions data
- n is the outward normal vector

From now on, the system of definition 2 will be the reference analytical problem for the development of numerical schemes.
To conclude, there exist other famous and useful models, such as the *Monodomain model*. But this is just a simplification of the Bidomain as in this case it is assumed that ϕ_i and ϕ_e are proportional. However, thanks to its simplicity, we often tested the code starting from the Monodomain implementation of the project [1] instead of analyzing directly the Bidomain.

1.4 A short discussion about the past works issue

As we have already introduced, our project initially aimed to continue and improve the work of a previous project ([7]).
Results obtained using unitary parameters, namely $\chi_m = \Sigma_i = \Sigma_e = C_m = k = \epsilon = \gamma = a = 1$, were actually quite satisfactory. Instead, the choice of more realistic/experimental values for the parameters (that are often very big or very small) caused bad consequences to the accuracy of the schemes or even to their stability. In particular, we observed that the choice of $C_m \approx 10^{-2}$ highly compromised the stability of the numerical schemes. This issue heavily limits the use of the code for research and/or experimental simulations as it guarantees convergence to the right solution only in few and non-realistic problems.

After a while, we realized that an inverted sign of the FitzHugh-Nagumo model formula occurred in [9].

This oversight was not only essential for the fidelity to the real phenomena but also crucial for the well-posedness of the problem.
We could give two motivations to reinforce this last statement: first of all, if we consider the well known study of Bourgault, Coudière, and Pierre ([4]), the conditions required for the well-posedness of the Bidomain problem are not satisfied if the sign is inverted (neither for the existence, hypothesis H4, nor for the uniqueness).
Secondly, suppose to discretize the Bidomain problem in time and to treat the nonlinear term semi-implicitly, as it will be done in the following sections. Then, if we fix the timestep and if data from the previous timestep are given, we achieve a linear problem that can be easily switched into a weak formulation. From this analysis, we could observe that if $V_m < a$ or $V_m > 1$ and C_m is sufficiently small, the associated bilinear form is not coercive.

This second motivation, even if not very formal and with a mixed approach, is particularly interesting since a few confirmations occurred during simulations, as shown for instance in figure 3.

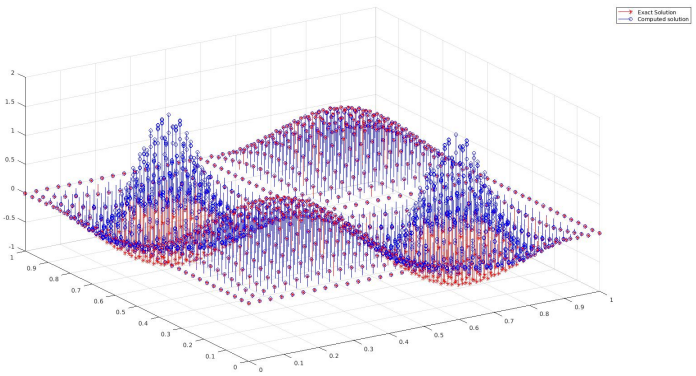

Figure 3: Comparison between exact and computed solution with inverted FitzHugh-Nagumo model: huge errors arise only when $Vm < 0$

In conclusion, the issue of the past works [1], [7] had not a numerical nature as expected, but instead an analytical origin due to an ill-posed problem.

2 Semi discretized numerical methods

2.1 DG discrete formulation

We have seen the Bidomain model in a complete form in definition 2. We now introduce a triangulation τ_h over Ω, with $\mathcal{F}_h = \mathcal{F}_h^I \cup \mathcal{F}_h^B$ set of the faces of the elements, which includes the internal and boundary faces respectively, and the DG space $V_h^k = \{v_h \in L^2(\Omega) : v_h|_{\mathcal{K}} \in \mathbb{P}^k(\mathcal{K}) \quad \forall \mathcal{K} \in \tau_h\}$, where k is the degree of the piecewise continuous polynomial. We obtain the semi discrete DG formulation:

For any time $t \in [0,T]$ find $\Phi_h(t) = [\phi_i^h(t), \phi_e^h(t)]^T \in [V_h^k]^2$ and $w_h(t) \in V_h^k$ such that

1.
$$\sum_{K\in\tau_h}\int_K \chi_m C_m \frac{\partial V_m^h}{\partial t} V_h dw + a_i(\phi_i^h, v_h) + \sum_{K\in\tau_h}\int_K \chi_m k(V_m^h - 1)(V_m^h - a)V_m^h v_h dw + \\ + \sum_{K\in\tau_h}\int_K \chi_m w_h v_h dw = (I_i^{ext}, v_h) \qquad \forall v_h \in V_h^p$$

2.
$$-\sum_{K\in\tau_h}\int_K \chi_m C_m \frac{\partial V_m^h}{\partial t} V_h dw + a_e(\phi_e^h, v_h) - \sum_{K\in\tau_h}\int_K \chi_m k(V_m^h - 1)(V_m^h - a)V_m^h v_h dw + \\ - \sum_{K\in\tau_h}\int_K \chi_m w_h v_h dw = (-I_e^{ext}, v_h) \qquad \forall v_h \in V_h^p$$

3.
$$\sum_{K\in\tau_h}\int_K \frac{\partial w_h}{\partial t} v_h dw = \sum_{k\in\tau_h}\int_K \epsilon(V_m^h - \gamma w_h) v_h dw \qquad \forall v_h \in V_h^p$$

where:

- $a_k(\phi_k^h, v_h) = \sum_{K\in\tau_h}\int_K (\Sigma_k \nabla_h \phi_k^h)\cdot\nabla_h v_h dw - \sum_{F\in\mathcal{F}_h^I}\int_F \{\{\Sigma_k \nabla_h \phi_k^h\}\}\cdot[[v_h]]d\sigma +$
$$- \delta \sum_{F\in\mathcal{F}_h^I}\int_F \{\{\Sigma_k \nabla_h v_h\}\}\cdot[[\phi_k^h]]d\sigma + \sum_{F\in\mathcal{F}_h^I}\int_F \gamma[[\phi_k^h]]\cdot[[v_h]]d\sigma \qquad k = i, e$$
- $(I_i^{ext}, v_h) = \sum_{K\in\tau_h}\int_K I_i^{ext} v_h dw + \int_{\partial w} b_i v_h d\sigma$
- $(-I_e^{ext}, v_h) = -\sum_{K\in\tau_h}\int_K I_e^{ext} v_h dw + \int_{\partial w} b_e v_h d\sigma$

Moreover, according to the choice of the coefficient δ, we can define:

- $\delta = 1$: Symmetric Interior Penalty method (SIP)
- $\delta = 0$: Incomplete Interior Penalty method (IIP)
- $\delta = -1$: Non Symmetric Interior Penalty method (NIP)

And $\gamma := \alpha \frac{k^2}{h}$ (*"Stabilization parameter"*), $\alpha \in \mathbb{R}$ to be chosen high enough.

To analyze the formulation more in details see [7].

2.2 Algebraic formulation

Taking $\{\varphi_j\}_{j=1}^{N_h}$ base of V_h^k, so that we can write

$$\Phi_h(t) = \begin{bmatrix} \phi_i^h(t) \\ \phi_e^h(t) \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{N_h} \phi_{i,j}(t)\varphi_j \\ \sum_{j=1}^{N_h} \phi_{e,j}(t)\varphi_j \end{bmatrix}$$

$$w_h(t) = \sum_{j=1}^{N_h} w_j(t)\varphi_j$$

$$V_m^h(t) = \sum_{j=1}^{N_h} V_{m,j}(t)\phi_j = \sum_{j=1}^{N_h} (\phi_{i,j}(t) - \phi_{e,j}(t))\varphi_j$$

Then, we introduce the matrices:

$$\left.\begin{array}{ll} (V_k)_{ij} & = \int_w \nabla\varphi_j \cdot \Sigma_k \nabla\varphi_i \\ (I_k^T)_{ij} & = \sum_{F\in F_h^I} \int_F \{\{\Sigma_k \nabla\varphi_j\}\} \cdot [[\varphi_i]] \\ (I_k)_{i,j} & = \sum_{F\in F_h^I} \int_F [[\varphi_j]] \cdot \{\{\Sigma_k \nabla\varphi_i\}\} \\ (S_k)_{i,j} & = \sum_{F\in F_h^I} \int_F \gamma_k [[\varphi_j]] \cdot [[\varphi_i]] \end{array}\right\} \quad \begin{array}{c} A_k = (V_k - I_k^T - \theta I_k + S_k) \\ k = i, e \end{array} \tag{1}$$

$$\gamma_k|_F = (n_F^T \Sigma_k \, n_F)\, \gamma, \quad n_F \text{ outward normal vector of F}$$

$$A_i \qquad \text{Intra-cellular stiffness matrix} \tag{2}$$

$$A_e \qquad \text{Extra-cellular stiffness matrix} \tag{3}$$

$$M_{ij} = \sum_{K\in\tau_h} \int_K \varphi_j\varphi_i \qquad \text{Mass matrix} \tag{4}$$

$$C(u_h)_{ij} = \sum_{K\in\tau_h} \int_K \chi_m k(u_h - 1)(u_h - a)\varphi_j\varphi_i \qquad \text{Non-linear matrix} \tag{5}$$

$$F_k = \begin{bmatrix} F_{i,k} \\ F_{e,k} \end{bmatrix} = \begin{bmatrix} \int_w I_i^{ext}\varphi_k - \sum_{F\in F_h^B}\int_F b_i\varphi_k \\ -\int_w I_e^{ext}\varphi_k - \sum_{F\in F_h^B}\int_F b_e\varphi_k \end{bmatrix} \tag{6}$$

Therefore, our semi-discrete algebraic formulation is:

find $\Phi_h(t) = [\phi_i^h(t), \phi_e^h(t)]^T \in [V_h^k]^2$ and $w_h(t) \in V_h^k$ for any $t \in (0;T]$ such that:

$$\begin{gathered} \chi_m Cm M\dot{V_m^h} + A_i\phi_i^h + C(V_m^h)V_m^h + \chi_m M w_h = F_i^h \\ -\chi_m Cm M\dot{V_m^h} + A_e\phi_e^h - C(V_m^h)V_m^h - \chi_m M w_h = F_e^h \\ M\dot{w}_h(t) = \epsilon M(V_m^h(t) - \gamma w_h(t)) \end{gathered} \tag{7}$$

Rewriting it with block matrices:

find $\Phi_h(t) = [\phi_i^h(t), \phi_e^h(t)]^T \in [V_h^k]^2$ and $w_h(t) \in V_h^k$ for any $t \in (0;T]$ such that:

$$\begin{gathered} \chi_m C_m \begin{bmatrix} M & -M \\ -M & M \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\phi}}_{\boldsymbol{i}}^{\boldsymbol{h}}(\boldsymbol{t}) \\ \dot{\boldsymbol{\phi}}_{\boldsymbol{e}}^{\boldsymbol{h}}(\boldsymbol{t}) \end{bmatrix} + \begin{bmatrix} A_i & 0 \\ 0 & A_e \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}_{\boldsymbol{i}}^{\boldsymbol{h}}(\boldsymbol{t}) \\ \boldsymbol{\phi}_{\boldsymbol{e}}^{\boldsymbol{h}}(\boldsymbol{t}) \end{bmatrix} + \\ \begin{bmatrix} C(V_m^h) & -C(V_m^h) \\ -C(V_m^h) & C(V_m^h) \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi}_{\boldsymbol{i}}^{\boldsymbol{h}}(\boldsymbol{t}) \\ \boldsymbol{\phi}_{\boldsymbol{e}}^{\boldsymbol{h}}(\boldsymbol{t}) \end{bmatrix} + \chi_m \begin{bmatrix} M & 0 \\ 0 & -M \end{bmatrix} \begin{bmatrix} w_h(t) \\ w_h(t) \end{bmatrix} = \begin{bmatrix} F_i^h \\ F_e^h \end{bmatrix} \\ \dot{w}_h(t) = \epsilon(V_m^h(t) - \gamma w_h(t)) \end{gathered} \tag{8}$$

3 Dubiner Basis

3.1 Analytical aspects

So far, we have described a general semi-discrete discontinuous formulation without examining which basis to use to generate the V_h^k space. Usually, the common choice consists in the classical hat functions from FEM, even if they need to be modified in order to be used in a discontinuous context. It is also one of the simplest choices, for this reason our provided code was initially implemented with this basis. However, the very novelty of this study is the adoption of a new kind of basis, completely different from the previous and commonly known as "*Dubiner Basis*" [6].
How we will soon see, the peculiarity of this family of functions is that it consists of orthogonal polynomials defined on the reference triangle

$$\hat{K} = \{(\xi,\eta) : \xi,\eta \geq 0, \xi+\eta \leq 1\} \tag{9}$$

and not on the reference square

$$\hat{Q} = \{(a,b) : -1 \leq a \leq 1, -1 \leq b \leq 1\} \tag{10}$$

Formally, if we consider the transformation from $\hat{Q}$ to $\hat{K}$

$$\xi := \frac{(1+a)(1-b)}{4}, \eta := \frac{(1+b)}{2} \tag{11}$$

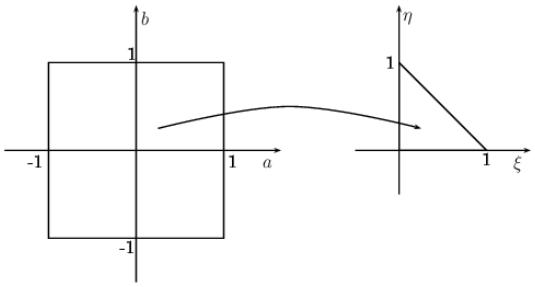

Figure 4: Transformation between the reference square to the reference triangle

the Dubiner basis is the transformation of a suitable basis initially defined on the reference square. This initial basis is simply obtained with a two dimensional modified tensor product of the Jacobi polynomials on the interval $(-1, 1)$.

Definition 3 (Jacobi polynomials)**.** *The Jacobi polynomials of coefficients* $\alpha, \beta \in \mathbb{R}$ *evaluated in* $z \in (-1, 1)$ *are:*

- $n = 0$

$$J_0^{\alpha,\beta}(z) = 1 \tag{12}$$

- $n = 1$

$$J_1^{\alpha,\beta}(z) = \frac{1}{2}(\alpha - \beta + (\alpha + \beta + 2) \cdot z); \tag{13}$$

- $n \geq 2$

$$\begin{aligned} J_n^{\alpha,\beta}(z) = \sum_{k=2}^{n} \Big[& \frac{(2k+\alpha+\beta-1)(\alpha^2-\beta^2)}{2k(k+\alpha+\beta)(2k+\alpha+\beta-2)} + \\ & \frac{(2k+\alpha+\beta-2)(2k+\alpha+\beta-1)(2k+\alpha\beta)}{2k(k+\alpha+\beta)(2k+\alpha+\beta-2)} J_{k-1}^{\alpha,\beta}(z) + \\ & - \frac{2(k+\alpha-1)(k+\beta-1)(2k+\alpha+\beta)}{2k(k+\alpha+\beta)(2k+\alpha+\beta-2)} J_{k-2}^{\alpha,\beta}(z) \Big] \end{aligned} \tag{14}$$

An important property of these polynomials is:

Proposition 1. $J_i^{\alpha,\beta}(\cdot)$ *is orthogonal w.r.t. the Jacobi weight* $w(x) = (1-x)^\alpha(1+x)^\beta$*:*

$$\int_{-1}^{1} (1-x)^\alpha (1+x)^\beta J_m^{\alpha,\beta} J_q^{\alpha,\beta}(x) dx = \frac{2}{2m+1} \delta_{mq} \tag{15}$$

Thanks to this definition, we can now define explicitly the Dubiner basis.

Definition 4 (Dubiner Basis)**.** *The Dubiner basis that generates the space* $\mathbb{P}^p(\hat{K})$ *of the polynomials of degree* p *over the reference triangle is the set of functions:*

$$\begin{aligned} &\phi_{ij} : \hat{K} \to \mathbb{R} \\ &\phi_{ij}(\xi,\eta) := c_{ij}(1-b)^j J_i^{0,0}(a) J_j^{2i+1,0}(b) = \\ &\qquad = c_{ij} 2^j (1-\eta)^j J_i^{0,0}(\frac{2\xi}{1-\eta} - 1) J_j^{2i+1,0}(2\eta - 1) \end{aligned} \tag{16}$$

for $i, j = 1, \ldots, p$ *and* $i + j \leq p$*, where*

$$c_{ij} := \sqrt{\frac{2(2i+1)(i+j+1)}{4^i}} \tag{17}$$

and $J_i^{\alpha,\beta}(\cdot)$ *is the i-th Jacobi polynomial*

As we have anticipated

Proposition 2. *The Dubiner basis is orthonormal in* $L^2(\hat{K})$ $\forall p$*:*

$$\int_{\hat{K}} \phi_{ij}(\xi,\eta)\phi_{mq}(\xi,\eta) d\xi d\eta = \delta_{im}\delta_{jq} \tag{18}$$

As a consequence, after we successfully implemented the code with Dubiner basis and computed the matrices, we obtained a diagonal mass matrix (figure 5)

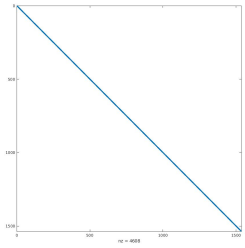

Figure 5: Non-zero elements in the mass matrix when adopting Dubiner basis

It is noteworthy to point out that transformation 11 is bijective, it can be inverted but it needs some care. The natural inverse would be:

$$a = \frac{2\xi}{1-\eta} - 1 \qquad b = 2\eta - 1 \tag{19}$$

that has already been used for the definition 4. However, it is not defined for $\eta = 1$, that means for the sole point $(0, 1)$ of the reference triangle. To solve this issue, it is enough to prolong the function with continuity to this special point. For the code implementation, it is suggested avoiding evaluations in the exact point or adding an *if* condition. We opted for the second solution.

In general, the orthogonality property implies some good numerical properties, not only the diagonalization of the mass matrix. For instance, in [2] interesting bounds for the conditional number can be viewed. For this reason, we opted for this choice aiming to improve the previous results, at least from the space discretization side.

On the other hand, there are also some difficulties arising when one chooses to abandon the familiar FEM basis. First of all, the coefficients of a discretized function has only *modal* meaning and they no more represent the *nodal* values of the function itself. This fact needs some extra work when one needs to switch from the continuous functions to the discretized functions and viceversa, as it will be shown in the paragraph 3.2. Secondly, one can notice that these functions are not boundary conditions friendly. What we mean is that, if compared to FEM basis, they have no particular properties on the boundary to let easily impose homogeneous boundary conditions. Thus, they should be again transformed, this time in a *boundary adapted* form. We address to [8] for a short description of this procedure. Fortunately, we do not need to set this transformation as in the discontinuous formulation boundary conditions (both Dirichlet and Neumann) are imposed only weakly. It means that the boundary conditions' choice does not imply the choice of the vectorial space as in continuous Galerkin. The discretized space is always the same, only some terms in the weak formulation have in case of need to be changed. For this reason, the match of Discontinuous Galerkin and Dubiner Basis results to be particularly successful.

To conclude, we refer to [10] for the transformation and the definition of the Dubiner Basis with tetrahedra, i.e. in dimension $n = 3$.

3.2 Implementation

Our code let the user to select which basis to adopt (FEM or Dubiner) and the order of polynomials until the order 3. We chose to call D_1, D_2, D_3 these 3 families of basis functions, thanks to the similarity to the P_1, P_2, P_3 finite element basis.
The starting point was the implementation of some functions to evaluate the Dubiner basis functions and their gradients in the quadrature points. We omit the

full code as it is not particular interesting: it barely follows the definitions of section 3.1 with the addition of some technicalities. These mentioned scripts are: `eval_jacobi_polynomial.m`, `basis_legendre_dubiner.m`, `evalshape_tria_dubiner.m`.

Moreover, some conditional statements and some extra methods (as the script `matrix2D_dubiner.m`) were added to let the user easily switch from one basis to another (simply and once via `dati.m`).

More interesting are instead the scripts `dubiner_to_fem.m` and `fem_to_dubiner.m`, used to convert the Dubiner modal coefficients of the vector solution to the nodal values of the approximated function and viceversa. For this reason, they deserve some further explanations.

3.2.1 Switch from the modal coefficients to the nodal values

One of the many advantages of the FEM basis is that there exists a bijection between the basis functions and some particular spacial points in such a way that the evaluation of a basis function in one of these points is equal to 1 only if that point is the one associated to the function, 0 otherwise:

$$\psi_i(x_j) = \delta_{ij} \tag{20}$$

Obviously, this property does not hold when we work with Dubiner basis. Indeed, these functions are not normalized on the mesh edges and they neither have an associated mesh point. This implies that the Dubiner coefficients of a function $u \in V_h^p$ are not the evaluation over these points of the discretized function itself. They have a completely different meaning, they are now *modal* values instead of being *nodal*. For this reason we introduced two new functions that best transform the coefficients of the solution w.r.t. FEM basis to the coefficients w.r.t. Dubiner basis and viceversa.

Consider an element $\mathcal{K} \in \tau_h$ and $\{\psi_i\}_{i=1}^p$,$\{\phi_j\}_{j=1}^q$ as, respectively, the set of FEM functions and the set of Dubiner functions with support in $\mathcal{K}$. In addition, consider as $\{\hat{u}_i\}_{i=1}^p$,$\{\tilde{u}_j\}_{j=1}^q$ as, respectively, the FEM and Dubiner coefficients of the solution.

Let us start from the transformation to the FEM coefficients. We now exploit the property 20, i.e. the coefficient $\hat{u}_i$ is nothing else but the evaluation of u_h on the i-th mesh point, then:

$$\hat{u}_i = \sum_{j=1}^{q} \tilde{u}_j \phi_j(x_i) \tag{21}$$

where x_i is the point associated to the ψ_i basis function.

Instead, to compute the coefficients conversely, we need to exploit the fact that the Dubiner Basis are L^2-orthonormal (proposition 2). We then need to compute a L^2 scalar product between the FEM discretized function and each Dubiner basis function. That means:

$$\tilde{u}_j = \int_{\mathcal{K}} u_h(x)\phi_j(x)\,dx = \int_{\mathcal{K}} \sum_{i=1}^{p} \hat{u}_i\psi_i(x)\phi_j(x)\,dx = \sum_{i=1}^{p} \Big(\int_{\mathcal{K}} \psi_i(x)\phi_j(x)\,dx \Big)\hat{u}_i \tag{22}$$

If the Dubiner functions are chosen as Galerkin basis, both the transformations are needed for the code implementation. Formula 21 is needed to plot and compute errors after the resolution of the system (otherwise solely Dubiner coefficients are useless). Formula 22 is instead needed to convert the FEM initial data u_0 into a vector of Dubiner coefficients before the resolution of the system.

In order to be rigorous, but also for the sake of simplicity, these transformations are implemented only from P_n to D_n, $n = 1, 2, 3$ and viceversa. With this choice, the two basis generate the same space V_h^n and then the transformation infers only on the coefficients and not on the function's properties. Otherwise, decreasing n would mean to lose significant information, while increasing n does not substantially improve the quality of the solution as it initially belonged to a lower order space. Moreover, choosing the same degree for P and D implies several semplifications, for instance the same number of local nodes (nln). For this reason, both p and q are actually replaced with nln in the code.

To conclude, formula 21 to get nodal values has then been implemented in the following lines from the `dubiner_to_fem.m` script.

```
function [u0] = dubiner_to_fem (uh, femregion, Data)

...
...
...

u0 = zeros(femregion.ndof,1);

% loop over all the elements
for ie = 1:femregion.ne

   % to get the global indexes for the nodes of ie
   index = (ie-1)*femregion.nln*ones(femregion.nln,1) + [1:femregion.nln]';

   for i = 1 : femregion.nln
       for j = 1: femregion.nln
```

```
          u0(index(i)) = u0(index(i)) +  uh(index(j))*phi(1,i,j);
        end
    end

end
```

On the other hand, the slightly more difficult formula 22 have been reproduced in `fem_to_dubiner.m` using *Gauss-Legendre-Lobatto* quadrature integration formulas. The main steps are the following:

```
function [u0] = fem_to_dubiner (uh, femregion, Data)

...
...
...

u0 = zeros(femregion.ndof,1);

% loop over all the elements
for ie = 1:femregion.ne

    % to get the global indexes for the nodes of ie
    index = (ie-1)*femregion.nln*ones(femregion.nln,1) + [1:femregion.nln]';
    % loop over local degrees of freedom
    for i = 1 : femregion.nln
        % loop over 2D quadrature points
        for k = 1:length(w_2D)
            uh_eval_k = 0;
            % loop to evaluate uh in a quadrature point
            for j = 1:femregion.nln
                uh_eval_k = uh_eval_k + uh(index(j))*phi_fem(1,k,j);
            end
            u0(index(i)) = u0(index(i)) + uh_eval_k*phi_dub(1,k,i).*w_2D(k);
        end
    end
end
```

4 Temporal discretization

The problem is time dependent: after the spacial discretization, we then need to divide the interval (0,T] into K subintervals $(t^k, t^k + 1]$ of length Δt such that $t^k = k\Delta t \quad \forall k = 0, \cdots, K-1$ assuming $V_m^k \approx V_m(t^k)$. We have developed, implemented and tested 3 different temporal strategies: *semi-implicit*, *Godunov operator-splitting* and *quasi-implicit operator-splitting*.

4.1 Semi-implicit method

One of the most famous and used temporal scheme for a non-linear problem such as the Bidomain is certainly the *Semi-Implicit* scheme [9]. The basic idea is to treat most of the terms implicitly while in fact treating the non-linear term semi-implicitly. Since the non-linear is cubic, the best choice is to treat only one of these V_m terms implicitly, in this way:

$$I_{ion}^{k+1} = k(V_m^k - a)(V_m^k - 1)V_m^{k+1} + w^{k+1}$$

at each time-step k (different from the k parameter). Moreover, the gating variable ODE is treated implicitly with the exception of the term V_m:

$$M\frac{w^{k+1} - w^k}{\Delta t} = \epsilon M(V_m^k - \gamma w^{k+1})$$

Therefore, we obtain:

$$\begin{gathered}
\chi_m C_m \begin{bmatrix} M & -M \\ -M & M \end{bmatrix} \begin{bmatrix} \frac{\boldsymbol{\phi_i^{k+1}} - \phi_i^k}{\boldsymbol{\Delta t}} \\ \frac{\boldsymbol{\phi_e^{k+1}} - \phi_e^k}{\boldsymbol{\Delta t}} \end{bmatrix} + \begin{bmatrix} A_i & 0 \\ 0 & A_e \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi_i^{k+1}} \\ \boldsymbol{\phi_e^{k+1}} \end{bmatrix} + \\
\begin{bmatrix} C(V_m^h) & -C(V_m^h) \\ -C(V_m^h) & C(V_m^h) \end{bmatrix} \begin{bmatrix} \boldsymbol{\phi_i^{k+1}} \\ \boldsymbol{\phi_e^{k+1}} \end{bmatrix} + \chi_m \begin{bmatrix} M & 0 \\ 0 & -M \end{bmatrix} \begin{bmatrix} w^{k+1} \\ w^{k+1} \end{bmatrix} = \begin{bmatrix} F_i^{k+1} \\ F_e^{k+1} \end{bmatrix} \\
M\frac{w^{k+1} - w^k}{\Delta t} = \epsilon M(V_m^k - \gamma w^{k+1})
\end{gathered} \tag{23}$$

reminding that $V_m^k = \phi_i^k - \phi_e^k$.

We can reformulate in this way:

$$\begin{gathered}
\left(\frac{\chi_m C_m}{\Delta t} \begin{bmatrix} M & -M \\ -M & M \end{bmatrix} + \begin{bmatrix} A_i & 0 \\ 0 & A_e \end{bmatrix} + \begin{bmatrix} C(V_m^k) & -C(V_m^k) \\ -C(V_m^k) & C(V_m^k) \end{bmatrix} \right) \begin{bmatrix} \boldsymbol{\phi_i^{k+1}} \\ \boldsymbol{\phi_e^{k+1}} \end{bmatrix} = \\
\begin{bmatrix} F_i^{k+1} \\ F_e^{k+1} \end{bmatrix} - \chi_m \begin{bmatrix} M & 0 \\ 0 & -M \end{bmatrix} \begin{bmatrix} w^{k+1} \\ w^{k+1} \end{bmatrix} + \frac{\chi_m C_m}{\Delta t} \begin{bmatrix} M & 0 \\ 0 & -M \end{bmatrix} \begin{bmatrix} V_m^k \\ V_m^k \end{bmatrix}
\end{gathered} \tag{24}$$

$$(\frac{1}{\Delta t} + \epsilon\gamma)Mw^{k+1} = \epsilon MV_m^k + \frac{M}{\Delta t}w^k \tag{25}$$

If we define:

- $B = \frac{\chi_m C_m}{\Delta t}\begin{bmatrix} M & -M \\ -M & M \end{bmatrix} + \begin{bmatrix} A_i & 0 \\ 0 & A_e \end{bmatrix}$
- $C_{nl}(V_m^k) = \begin{bmatrix} C(V_m^h) & -C(V_m^h) \\ -C(V_m^h) & C(V_m^h) \end{bmatrix}$
- $r^{k+1} = \begin{bmatrix} F_i^{k+1} \\ F_e^{k+1} \end{bmatrix} - \chi_m \begin{bmatrix} M & 0 \\ 0 & -M \end{bmatrix}\begin{bmatrix} w^{k+1} \\ w^{k+1} \end{bmatrix} + \frac{\chi_m C_m}{\Delta t}\begin{bmatrix} M & -M \\ -M & M \end{bmatrix}\begin{bmatrix} \phi_i^k \\ \phi_e^k \end{bmatrix}$

we get the final system:

find $\Phi^{k+1} = [\phi_i^{k+1}\phi_e^{k+1}]^T$ and w^{k+1} $\forall k = 0, \cdots, T-1$ such that:

$$\begin{cases} (\frac{1}{\Delta t} + \epsilon\gamma)Mw^{k+1} = \epsilon MV_m^k + \frac{M}{\Delta t}w^k \\ (B + C_{nl}(\Phi^k))\Phi^{k+1} = r^{k+1} \end{cases} \tag{26}$$

4.2 Godunov operator-splitting

The main feature of a general operator-splitting method is the division of the problem into two different problems to be solved sequentially. This is possible and justified when the original functional operator L is splitted into 2 different operators such that $L(u) = L_1(u) + L_2(u)$. Two operator-splitting methods have been implemented, the first is of *Godunov* type and a detailed study together with its properties can be found in [11]. The formulation becomes:

find $\hat{V}_m^{k+1}, \phi_i^{k+1}, \phi_e^{k+1}, w^{k+1}$ such that:

I
$$\begin{cases} \chi_m C_m M\frac{\hat{V}_m^{k+1} - V_m^k}{\Delta t} + C(V_m^k)V_m^k + \chi_m M w^k = 0 \\ \frac{w^{k+1} - w^k}{\Delta t} = \epsilon(V_m^k - \gamma w^k) \end{cases}$$

II
$$\begin{cases} \chi_m C_m M\frac{V_m^{k+1} - \hat{V}_m^{k+1}}{\Delta t} + A_i\phi_i^{k+1} = F_i^{k+1} \\ -\chi_m C_m M\frac{V_m^{k+1} - \hat{V}_m^{k+1}}{\Delta t} + A_e\phi_e^{k+1} = F_e^{k+1} \end{cases}$$

Putting in a unique system:

$$\begin{cases} \chi_m C_m M \frac{V_m^{k+1}-V_m^k}{\Delta t} + C(V_m^k)V_m^k + \chi_m M w^k + A_i \phi_i^{k+1} = F_i^{k+1} \\ \chi_m C_m M \frac{V_m^{k+1}-V_m^k}{\Delta t} + C(V_m^k)V_m^k + \chi_m M w^k - A_e \Phi_e^{k+1} = -F_e^{k+1} \\ w^{k+1} = (1-\epsilon\gamma\Delta t)w^k + \epsilon\Delta t V_m^k \end{cases} \tag{27}$$

The equations in the system 27 can be rewritten as:

$$\begin{cases} \left(\frac{\chi_m C_m}{\Delta t}M + A_i\right)\phi_i^{k+1} - \frac{\chi_m C_m}{\Delta t}M\phi_e^{k+1} = F_i^{k+1} - \chi_m M w^k + \left(\frac{\chi_m C_m}{\Delta t}M - C(V_m^k)\right)V_m^k \\ \frac{\chi_m C_m}{\Delta t}M\phi_i^{n+1} - \left(\frac{\chi_m C_m}{\Delta t}M + A_e\right)\phi_e^{k+1} = -F_e^{k+1} - \chi_m M w^k + \left(\frac{\chi_m C_m}{\Delta t}M - C(V_m^k)\right)V_m^k \\ w^{k+1} = (1-\epsilon\gamma\Delta t)w^k + \epsilon\Delta t V_m^k \end{cases}$$

In the end, one can compute $\Phi^{k+1} = [\phi_i^{k+1}\phi_e^{k+1}]^T$ and $w^{k+1} \quad \forall k = 0, \cdots, K-1$ from:

$$\begin{cases} \left(\frac{\chi_m C_m}{\Delta t}\begin{bmatrix} M & -M \\ M & -M \end{bmatrix} + \begin{bmatrix} A_i & 0 \\ 0 & -A_e \end{bmatrix}\right)\begin{bmatrix} \boldsymbol{\phi_i^{k+1}} \\ \boldsymbol{\phi_e^{k+1}} \end{bmatrix} = \begin{bmatrix} F_i^{k+1} \\ -F_e^{k+1} \end{bmatrix} + \\ -\chi_m \begin{bmatrix} M & 0 \\ 0 & M \end{bmatrix}\begin{bmatrix} w^k \\ w^k \end{bmatrix} + \left(\frac{\chi_m C_m}{\Delta t}\begin{bmatrix} M & 0 \\ 0 & M \end{bmatrix} - \begin{bmatrix} C(V_m^k) & 0 \\ 0 & C(V_m^k) \end{bmatrix}\right)\begin{bmatrix} V_m^k \\ V_m^k \end{bmatrix} \\ \\ w^{k+1} = (1-\epsilon\gamma\Delta t)w^k + \epsilon\Delta t V_m^k \end{cases} \tag{28}$$

4.2.1 Implementation

```
ZERO = sparse(ll,ll);
MASS = (ChiM*Cm/dt)*[M, -M; M -M];
MASSW = ChiM*[M, ZERO; ZERO, M];

for t=dt:dt:T

fi = assemble_rhs_i(femregion,neighbour,Data,t);
fe = assemble_rhs_e(femregion,neighbour,Data,t);
f1 = cat(1, fi, -fe);

[C] = assemble_nonlinear(femregion,Data,Vm0);

w1 = (1 -epsilon*gamma*dt)*w0 + epsilon*dt*Vm0;
B = MASS + [Ai, ZERO; ZERO, -Ae];
r = -MASSW*[w0;w0] + ((Cm/dt)*MASSW - [C, ZERO; ZERO, C])
*[Vm0;Vm0] + f1;

Vm0 = u1(1:ll) - u1(ll+1:end);
```

```
u0 = u1;
w0 = w1;
end
```

4.3 Quasi-implicit operator-splitting

The aim of a quasi-implicit operator splitting is to treat implicitly all the terms except the cubic one. Even if it cannot be defined as a full implicit method, we hope to achieve more stability if compared to the previous Godunov-kind scheme.

$$\text{find } \tilde{V}_m^{k+1}, \phi_i^{k+1}, \phi_e^{k+1}, w^{k+1} \text{ such that:}$$

I
$$\begin{cases} \chi_m C_m M \frac{\tilde{V}_m^{k+1} - V_m^k}{\Delta t} + C(V_m^k) V_m^{k+1} + \chi_m M w^{k+1} = 0 \\ \frac{w^{k+1} - w^k}{\Delta t} = \epsilon(V_m^{k+1} - \gamma w^{k+1}) \end{cases}$$

II
$$\begin{cases} \chi_m C_m M \frac{V_m^{k+1} - \tilde{V}_m^{k+1}}{\Delta t} + A_i \phi_i^{k+1} = F_i^{k+1} \\ -\chi_m C_m M \frac{V_m^{k+1} - \tilde{V}_m^{k+1}}{\Delta t} + A_e \phi_e^{k+1} = F_e^{k+1} \end{cases}$$

Putting into a unique system:

$$\begin{cases} \chi_m C_m M \frac{V_m^{k+1} - V_m^k}{\Delta t} + C(V_m^k) V_m^{k+1} + \chi_m M w^{k+1} + A_i \phi_i^{k+1} = F_i^{k+1} \\ \chi_m C_m M \frac{V_m^{k+1} - V_m^k}{\Delta t} + C(V_m^k) V_m^{k+1} + \chi_m M w^{k+1} - A_e \phi_e^{k+1} = -F_e^{k+1} \\ \frac{w^{k+1} - w^k}{\Delta t} = \epsilon(V_m^{k+1} - \gamma w^{k+1}) \end{cases} \tag{29}$$

If we define:

- $Q_k = \frac{\chi_m C_m}{\Delta t} M + C(V_m^k) + \frac{\epsilon \chi_m \Delta t}{1 + \epsilon \gamma \Delta t} M$
- $R_k = \frac{\chi_m C_m}{\Delta t} M V_m^k - \frac{\chi_m}{1 + \epsilon \gamma \Delta t} M w^k$

the equations in the system 29 can be written as:

1.
$$\chi_m C_m M \frac{\phi_i^{k+1} - \phi_e^{k+1} - V_m^k}{\Delta t} + C(V_m^k)(\phi_i^{k+1} - \phi_e^{k+1}) + \chi_m M \left(\frac{w^k + \epsilon \Delta t (\phi_i^{k+1} - \phi_e^{k+1})}{1 + \epsilon \gamma \Delta t} \right) +$$
$$+ A_i \phi_i^{k+1} = F_i^{k+1}$$
$$\Rightarrow \quad (Q_k + A_i)\phi_i^{k+1} - Q_k \phi_e^{k+1} = R_k + F_i^{k+1}$$

2.

$$\chi_m C_m M \frac{\phi_i^{k+1} - \phi_e^{k+1} - V_m^k}{\Delta t} + \cdot C(V_m^k)(\phi_i^{k+1} - \phi_e^{k+1}) + \chi_m M \left(\frac{w^k + \epsilon \Delta t (\phi_i^{k+1} - \phi_e^{k+1})}{1 + \epsilon \gamma \Delta t} \right) +$$

$$-A_e \phi_e^{k+1} = -F_e^{k+1}$$

$$\Rightarrow \quad Q_k \phi_i^{k+1} - (Q_k + A_e)\phi_e^{k+1} = R_k - F_e^{k+1}$$

3.

$$w^{k+1} = \frac{w^k + \epsilon \Delta t (\phi_i^{k+1} - \phi_e^{k+1})}{1 + \epsilon \gamma \Delta t}$$

The final system becomes:

find $\Phi^{k+1} = [\phi_i^{k+1} \phi_e^{k+1}]^T$ and $w^{k+1} \qquad \forall k = 0, \cdots, K-1$ such that:

$$\begin{cases} \left(\begin{bmatrix} Q_k & -Q_k \\ Q_k & -Q_k \end{bmatrix} + \begin{bmatrix} A_i & 0 \\ 0 & -A_e \end{bmatrix} \right) \begin{bmatrix} \boldsymbol{\phi_i^{k+1}} \\ \boldsymbol{\phi_e^{k+1}} \end{bmatrix} = \begin{bmatrix} R_k \\ R_k \end{bmatrix} + \begin{bmatrix} F_i^{k+1} \\ -F_e^{k+1} \end{bmatrix} \\ \\ w^{k+1} = \dfrac{w^k + \epsilon \Delta t (\phi_i^{k+1} - \phi_e^{k+1})}{1 + \epsilon \gamma \Delta t} \end{cases} \tag{30}$$

4.3.1 Implementation

```
ZERO = sparse(ll,ll);

   for t=dt:dt:T

        [C] = assemble_nonlinear(femregion,Data,Vm0);
         Q  = (ChiM*Cm/dt)*M + C - (epsilon*ChiM*dt)/(1+epsilon*gamma*dt)*M;
         R  = (ChiM*Cm/dt)*M*Vm0 - (ChiM)/(1+epsilon*gamma*dt)*M*w0;

        fi = assemble_rhs_i(femregion,neighbour,Data,t);
        fe = assemble_rhs_e(femregion,neighbour,Data,t);
        f1 = cat(1, fi, -fe);

        B = [Q, -Q; Q, -Q] + [Ai, ZERO; ZERO, -Ae];
        r = [R;R] + f1;

        u1 = B \ r;

        Vm1 = u1(1:ll)-u1(ll+1:end);
```

```
    w1 = (w0 + epsilon*dt*Vm1)/(1+epsilon*gamma*dt);

    f0 = f1;
    Vm0 = u1(1:ll) - u1(ll+1:end);
    u0 = u1;
    w0 = w1;
end
```

References

[1] F. Andreotti and D. Uboldi. *Discontinuous Galerkin approximation of the monodomain problem*. Politecnico di Milano, 2021.

[2] P. F. Antonietti and P. Houston. "A Class of Domain decomposition Preconditioners for hp-Discontinuous Galerkin Finite Element Methods". In: *Journal of Scientific Computing* 46 (2011), pp. 124–149.

[3] M. Bagnara. *The Inverse Potential Problem of Electrocardiography Regularized with Optimal Control*. Politecnico di Milano, 2020.

[4] Y. Bourgault, Y. Coudière, and C. Pierre. "Existence and uniqueness of the solution for the bidomain model used in cardiac electrophysiology". In: *Nonlinear Analysis: Real World Applications* 10 (2009), pp. 458–482.

[5] P. Colli Franzone, L. F. Pavarino, and S. Scacchi. *Mathematical Cardiac Electrophysiology*. Cham: Springer-Verlag, 2014.

[6] M. Dubiner. "Spectral Methods on Triangles and Other Domains". In: *Journal of Scientific Computing* 6 (1991), pp. 345–390.

[7] L. Marta and M. Perego. *Discontinuous Galerkin approximation of the bidomain system for cardiac electrophysiology*. Politecnico di Milano, 2021.

[8] A. Quarteroni. *Modellistica Numerica per Problemi Differenziali*. Milan: Springer-Verlag, 2016.

[9] A. Quarteroni, A. Manzoni, and C. Vergara. "The cardiovascular system: Mathematical modelling, numerical algorithms and clinical applications". In: *Acta Numerica* (2017), pp. 365–590.

[10] S. J. Sherwin and G. E. Karniadakis. "A new triangular and tetrahedral basis for high-order finite element methods". In: *Internation journal for numerical methods in engineering* 38 (1995), pp. 3775–3802.

[11] R. Spiteri and S. Torabi Ziaratgahi. "Operator splitting for the bidomain model revisited". In: *Journal of Computational and applied Mathematics* 296 (2016), pp. 550–563.