



POLITECNICO
MILANO 1863

COURSE OF NUMERICAL ANALYSIS FOR PARTIAL DIFFERENTIAL EQUATIONS

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 | 2 |
| 1.1 | Abstract | 2 |
| 1.2 | The physical problem | 2 |
| 1.3 | Mathematical models | 3 |
| 1.4 | Previous project | 5 |
| 1.4.1 | The main error | 5 |
| 2 | Semi discretized numerical methods | 6 |
| 2.1 | DG discrete formulation | 6 |
| 2.2 | Algebraic formulation | 7 |
| 3 | Dubiner Basis | 8 |
| 3.1 | Implementation | 10 |
| 3.1.1 | matrix2D_dubiner.m | 10 |
| 3.1.2 | dubiner_to_fem.m and fem_to_dubiner.m | 11 |
| 4 | Totally discretized numerical methods | 14 |
| 4.1 | Semi implicit method | 14 |
| 4.1.1 | Results | 15 |
| 4.2 | Quasi implicit operator splitting | 15 |
| 4.2.1 | Implementation | 17 |
| 4.2.2 | Results | 18 |
| 4.3 | Godunov operator splitting | 18 |
| 4.3.1 | Implementation | 19 |
| 4.3.2 | Results | 19 |

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], [6]). In particular, the very goal of this project is to improve the results obtained in [6] (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 very briefly since it has already been widely shown in the previous project (Marta and Perego). For a more complete explanation, we instead refer to [7].

The mechanical contraction and expansion of human heart has its origin in the *electrical activation* of the cardiac cells. At every heart-beat, myocytes are activated and deactivated following a characteristic electrical cycle (fig 1).

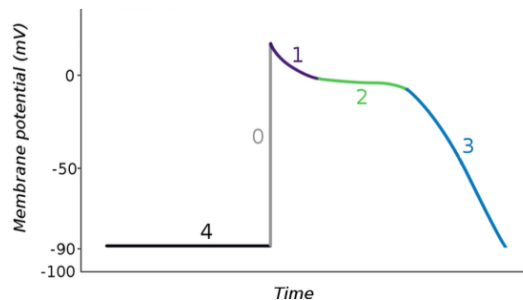


Figure 1: Membrane potential in function of time (one cardiac cycle)

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. Ca^{2+} , Na^{+} , K^{+}). From a macroscopical point of view, instead, one can think about 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 [7] for more details and/or [5] for the complete passages).

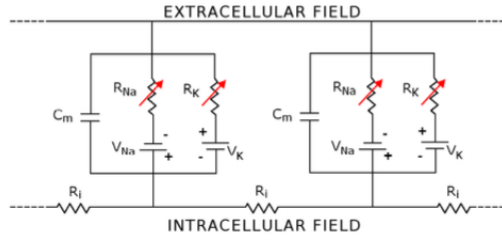


Figure 2: Simplified circuit to model the intracellular and extracellular potentials dynamics

The general formulation is then:

Definition 1 (Bidomain model).

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

where:

- ϕ_i, ϕ_e are the *Intracellular and Extracellular Potentials* (unknowns)
- $V_m = \phi_i - \phi_e$ is the *Trans-membrane Potential*
- $\chi_m, C_m, \Sigma_i, \Sigma_e$ are known constants
- I_i^{ext}, I_e^{ext} are applied currents
- I_{ion} is the *Ionic Current*

- w_{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).

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

where:

- 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 Previous project

As we have already said, our project aims to continue the work of a previous project, Marta and Perego ([6]). Their goal was to implement a Discontinuous Galerkin method for the bidomain problem.

Their result using unitary parameters, namely $\chi_m = 1, \Sigma_i = 1, \Sigma_e = 1, C_m = 1, k = 1, \epsilon = 1, \gamma = 1$ and $a = 1$, were quite good, but the problem comes when the parameters utilized are more realistic, as $\chi_m = 10^5, \Sigma_i = 0.12, \Sigma_e = 0.12, C_m = 0.01, k = 19.5, \epsilon = 1.2, \gamma = 0.1$ and $a = 0.013$, indeed we can observe in figure 3 the presence of instabilities for the solution of ϕ_i and ϕ_e .

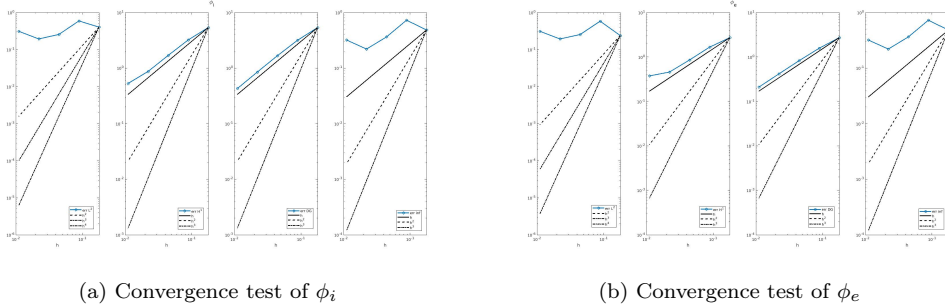


Figure 3: Convergence test using physiological parameters

This result limits the use of the code, because it cannot be used to study physiological cases.

1.4.1 The main error

We have done many changes in their code, but the main one regards the implementation of the very own model. Indeed, in the paper ([7]) we can find the equation for the Fitz-Hugh Nagumo model for the ionic current $I_{ion}(V_m, w) = -kV_m(V_m - a)(V_m - 1) - w$, having the model:

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

Instead if we see the most important paper regarding this model, [4] of Bourgault, Coudière, and Pierre, the equations of the bidomain model are the same, but the Fitz-Hugh Nagumo model has opposite sign, namely $I_{ion}(V_m, w) = kV_m(V_m - a)(V_m - 1) + w$.

This error was the principal contributor to the instabilities of the results of ([6]), because in some cases there wasn't coercivity of the bilinear form and

so the stability wasn't always assured.

Whereas in ([4]) the well posedness of the problem, using the right Fitz-Hugh Nagumo equation, is proved.

2 Semi discretized numerical methods

2.1 DG discrete formulation

We have seen the bidomain model in a complete form in definition 2. We introduce now a triangulation τ_h over w , 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 $V_h^k = \{v_h \in L^2 : v_h|_{\mathcal{K}} \in \mathbb{P}^k(\mathcal{K}) \quad \forall \mathcal{K} \in \tau_h\}$, where k is the degree of the piecewise continuous polynomial.

We obtain the semi discrete DG formulation:

$$\begin{aligned}
& \text{For any time } t \in [0, T] \text{ find } \Phi_h(t) = [\phi_i^h(t), \phi_e^h(t)]^T \in [V_h^k]^2 \text{ and } w_h(t) \in V_h^k : \\
& \sum_{K \in \tau_h} \int_K \chi_m C_m \frac{\partial V_m^h}{\partial t} V_h dw + a_i(\phi_i^h, v_h) + \sum_{K \in \tau_h} \int_K \chi_m k (V_m^h - 1)(V_m^h - a) V_m^h v_h dw + \\
& \quad + \sum_{K \in \tau_h} \int_K \chi_m w_h v_h dw = (I_i^{ext}, v_h) \quad \forall v_h \in V_h^p \\
& - \sum_{K \in \tau_h} \int_K \chi_m C_m \frac{\partial V_m^h}{\partial t} V_h dw + a_e(\phi_e^h, v_h) - \sum_{K \in \tau_h} \int_K \chi_m k (V_m^h - 1)(V_m^h - a) V_m^h v_h dw + \\
& \quad - \sum_{K \in \tau_h} \int_K \chi_m w_h v_h dw = (-I_e^{ext}, v_h) \quad \forall v_h \in V_h^p \\
& \sum_{K \in \tau_h} \int_K \frac{\partial w_h}{\partial t} v_h dw = \sum_{k \in \tau_h} \int_K \epsilon (V_m^h - \gamma w_h) v_h dw \quad \forall v_h \in V_h^p
\end{aligned}$$

where:

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

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)

To see in more detail the procedure see [6].

2.2 Algebraic formulation

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

$$\begin{aligned}\Phi_h(t) &= \begin{bmatrix} \phi_i^h(t) \\ \phi_e^h(t) \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{N_h} \phi_{i,j}(t) \varphi_j \\ \sum_{j=1}^{N_h} \phi_{e,j}(t) \varphi_j \end{bmatrix} \\ w_h(t) &= \sum_{j=1}^{N_h} w_j(t) \varphi_j \\ V_m^h(t) &= \sum_{j=1}^{N_h} V_{m,j}(t) \phi_j = \sum_{j=1}^{N_h} (\phi_{i,j}(t) - \phi_{e,j}(t)) \varphi_j\end{aligned}$$

Also we can introduce the matrices:

$$\left. \begin{aligned} V_{ij} &= \int_w \nabla \varphi_j \cdot \nabla \varphi_i \\ I_{i,j}^T &= \sum_{F \in F_h^I} \int_F \{ \{ \nabla \varphi_j \} \} \cdot [[\varphi_i]] \\ I_{i,j} &= \sum_{F \in F_h^I} \int_F [[\varphi_j]] \cdot \{ \{ \nabla \varphi_i \} \} \\ S_{i,j} &= \sum_{F \in F_h^I} \int_F \gamma [[\varphi_j]] \cdot [[\varphi_i]] \end{aligned} \right\} A = (V - I^T - \theta I + S) \quad (1)$$

$$A_i = \Sigma_i A \quad \text{Intra-cellular stiffness matrix} \quad (2)$$

$$A_e = \Sigma_e A \quad \text{Extra-cellular stiffness matrix} \quad (3)$$

$$M_{ij} = \sum_{K \in \tau_h} \int_K \text{Mass matrix} \varphi_j \varphi_i \quad (4)$$

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

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

Therefore our 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:

$$\begin{aligned} \chi_m C_m M \dot{V}_m^h + A_i \dot{\phi}_i^h + C(V_m^h) V_m^h + \chi_m M w_h &= F_i^h \\ -\chi_m C_m M \dot{V}_m^h + A_e \dot{\phi}_e^h - C(V_m^h) V_m^h - \chi_m M w_h &= F_e^h \\ M \dot{w}_h(t) &= \epsilon M (V_m^h(t) - \gamma w_h(t)) \end{aligned} \quad (7)$$

Rewrite it with the block matrix and assuming that M is non singular:
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:

$$\begin{aligned} \chi_m C_m \begin{bmatrix} M & -M \\ -M & M \end{bmatrix} \begin{bmatrix} \dot{\phi}_i^h(t) \\ \dot{\phi}_e^h(t) \end{bmatrix} + \begin{bmatrix} A_i & 0 \\ 0 & A_e \end{bmatrix} \begin{bmatrix} \phi_i^h(t) \\ \phi_e^h(t) \end{bmatrix} + \\ \begin{bmatrix} C(V_m^h) & -C(V_m^h) \\ -C(V_m^h) & C(V_m^h) \end{bmatrix} \begin{bmatrix} \phi_i^h(t) \\ \phi_e^h(t) \end{bmatrix} + \chi_m \begin{bmatrix} M & 0 \\ 0 & -M \end{bmatrix} \begin{bmatrix} w_h(t) \\ w_h(t) \end{bmatrix} &= \begin{bmatrix} F_i^h \\ F_e^h \end{bmatrix} \\ M \dot{w}_h(t) &= \epsilon (V_m^h(t) - \gamma w_h(t)) \end{aligned} \quad (8)$$

3 Dubiner Basis

We implemented another type of basis, the Dubiner basis, instead of the classic FEM basis. The most popular orthonormal basis on the reference triangle

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

is the Dubiner polynomial basis [2]. We consider the transformation in figure 4 between the reference square

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

and the reference triangle given by

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

The Dubiner basis is then constructed by a generalized tensor product of the Jacobi polynomials on the interval $(-1, 1)$ to form a basis on the reference square, which is then transformed by the above “collapsing” mapping to a basis on the reference triangle.

Definition 3 (Jacobi polynomials). *The Jacobi polynomials evaluated in $z \in \mathbb{R}^n$ are:*

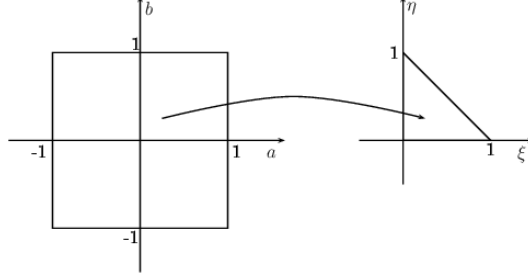


Figure 4: Transformation between the reference square to the reference triangle

– $n = 0$

$$J_0^{\alpha, \beta}(z) = \overbrace{[1 \quad 1 \quad \dots \quad 1]}^{n \text{ times}} \quad (12)$$

– $n = 1$

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

– $n \geq 2$

$$J_n^{\alpha, \beta}(z) = \sum_{k=2}^n \left[\frac{(2k + \alpha + \beta - 1)(\alpha^2 - \beta^2)}{2k(k + \alpha + \beta)(2k + \alpha + \beta - 2)} + \frac{(2k + \alpha + \beta - 2)(2k + \alpha + \beta - 1)(2k + \alpha\beta)}{2k(k + \alpha + \beta)(2k + \alpha + \beta - 2)} J_{k-1}^{\alpha, \beta}(z) + \frac{2(k + \alpha - 1)(k + \beta - 1)(2k + \alpha + \beta)}{2k(k + \alpha + \beta)(2k + \alpha + \beta - 2)} J_{k-2}^{\alpha, \beta}(z) \right] \quad (14)$$

An important property of this polynomials is:

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

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

Thanks to this definition, we can now look more accurately the formula of the Dubiner basis

Definition 4 (Dubiner Basis).

$$\begin{aligned} \phi_{ij}(\xi, \eta) &:= c_{ij}(1 - b)^j J_i^{0,0}(a) J_j^{2i+1,0}(b) = \\ &= c_{ij} 2^j (1 - \eta)^j J_i^{0,0}\left(\frac{2\xi}{1 - \eta} - 1\right) J_j^{2i+1,0}(2\eta - 1) \end{aligned} \quad (16)$$

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

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

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

We note that

Proposition 2. $\phi_{ij}(\xi, \eta)$ is orthogonal w.r.t. Legendre internal product:

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

Indeed, we obtain a mass matrix diagonal as we can see in figure 5

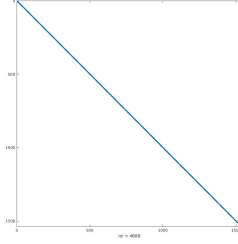


Figure 5: Elements non zero in the mass matrix w.r.t Dubiner basis

3.1 Implementation

Some of the codes that we used were written by professor Antonietti as `eval_jacobi_polynomial.m`, used to evaluate the Jacobi polynomial $J_n^{\alpha, \beta}$ in the vector `z`, `basis_legendre_dubiner.m`, that generates the basis functions of legendre Qp-Pp on the reference element $[-1, 1] \times [-1, 1]$, and `evalshape_tria_dubiner.m`, it creates the Dubiner basis (`dphiq`) and its gradient (`Grad`) on the volume terms and on the edges (`B_edge` and `G_edge`). So we used these ones as starting point for the functions that we needed.

Other simply changes in the code of (??), that we won't cover in this relation, were made in order to call the right function according to basis we have chosen (Dubiner or FEM). (METTERE CODICE DI QUESTE FUNZIONI?)

3.1.1 matrix2D_dubiner.m

At first we implemented a function that assembles the matrices that we have defined in (1): METTERE CODICE?

3.1.2 dubiner_to_fem.m and fem_to_dubiner.m

One of the many advantages of the FEM basis is that the evaluation of a basis function in a point of the mesh is equal to 1 only if that point is the one associated to the basis, 0 otherwise:

$$\psi_i(x_j) = \delta_{ij} \quad (19)$$

This property cannot be satisfied by Dubiner basis (although other good properties hold in this case, for instance regularity and especially *orthogonality*). Indeed these basis have not localized support and they are neither normalized on the mesh edges. This means that the coefficients of the solution of the Dubiner system are *not* the evaluation over the mesh 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 19, i.e. the coefficient \hat{u}_i is nothing else but the evaluation of u_h on the i -th mesh point, then:

$$\hat{u}_i = \sum_{j=1}^q \tilde{u}_j \phi_j(x_i) \quad (20)$$

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 (property obtained thanks to 1). We then need to compute a L^2 scalar product between the FEM discretized function and each Dubiner basis function. That means:

$$\tilde{u}_j = \int_{\mathcal{K}} u_h(x) \phi_j(x) dx = \int_{\mathcal{K}} \sum_{i=1}^p \hat{u}_i \psi_i(x) \phi_j(x) dx = \sum_{i=1}^p \left(\int_{\mathcal{K}} \psi_i(x) \phi_j(x) dx \right) \hat{u}_i \quad (21)$$

If the Dubiner functions are chosen as Galerkin basis, both the transformations are needed for the code implementation. Formula 20 is needed to plot

and compute errors after the resolution of the system (otherwise solely Dubiner coefficients are useless). Formula 21 is instead needed to convert the FEM initial data u_0 into a vector of Dubiner coefficients before the resolution of the system.

For the sake of both simplicity and logic, we have decided to implement these transformations only from P_n to D_n , $n = 1, 2, 3$ and viceversa. Indeed, from one side, the degree of FEM is here less "important" since it contributes only to the number of points to which evaluate the computed solution. Then, increasing n for P does not substantially improve the quality of the solution. On the other side, choosing the same degree for P and D means having the same number of local nodes (nln). For this reason, both p and q are replaced with nln in the code. Finally, we see the first transformation associated to 20:

```
function [u0] = dubiner_to_fem (uh, femregion, Data)
deg=sscanf(Data.fem(2:end), '%f');
s=0;

% define the coordinates of the degree of freedom in the reference triangle
% adn square
if (deg==1) %D1
    a    = [-1; 1; -1];
    b    = [-1; -1; 1];

elseif (deg ==2) %D2
    a    = [-1; 0; 1; 1; -1; -1];
    b    = [-1; -1; -1; 0; 1; 0];

elseif (deg==3) %D3
    a    = [-1; -0.5; 0.5; 1; 1; 1; -1; -1; -1; 0];
    b    = [-1; -1; -1; -1; -0.5; 0.5; 1; 0.5; -0.5; -1/3];
end

csi=(1+a).*(1-b)/4;
eta=(1+b)/2;

% evaluate the Dubiner basis on these points
for j=0:(deg)
    for i=0:(deg)
        if (i+j) <= deg
```

```

        s=s+1;
        [pi] = eval_jacobi_polynomial(i,0,0,a);
        [pj] = eval_jacobi_polynomial(j,2.*i+1,0,b);
        cij=sqrt((2.*i +1).*2.*(i+j+1)./4.^i);
        phi(1,:,s)=cij.*(2.^i).*((1-eta).^i).*pi.*pj;
    end
end
end

u0 = zeros(femregion.ndof,1);

for ie = 1:femregion.ne

    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
end

```

And the second one associated to 21:

```

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

% quadrature nodes and weights for integrals
[nodes_1D, ~, nodes_2D, w_2D] = quadrature(Data.nqn);

% evaluation of shape functions on quadrature point both on FEM basis and
% Dubiner basis
[shape_basis] = basis_legendre_dubiner(femregion.fem);
[phi_dub, ~, ~, ~] = evalshape_tria_dubiner(shape_basis,nodes_2D, nodes_1D,
Data.nqn,femregion.nln);
[shape_basis] = basis_lagrange(append("P", femregion.fem(2)));
[phi_fem, ~, ~, ~] = evalshape(shape_basis,nodes_2D,nodes_1D,femregion.nln);

u0 = zeros(femregion.ndof,1);

for ie = 1:femregion.ne

```

```

index = (ie-1)*femregion.nln*ones(femregion.nln,1) + [1:femregion.nln]';
for i = 1 : femregion.nln
    for k = 1:length(w_2D)
        uh_eval_k = 0;
        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
end

```

4 Totally discretized numerical methods

The system is time dependent, so we divide the interval $(0, T]$ into K subintervals $(t^k, t^k + 1]$ of length Δt such that $t^k = k\Delta t \quad \forall k = 0, \dots, K-1$ and so we consider $V_m^k \approx V_m(t^k)$.

4.1 Semi implicit method

At first, we consider an implicit method for the time discretization, meanwhile for the non linear contribution of I^{ion} a semi implicit method, indeed we think about it as a cubic function, so we treat explicitly the quadratic term and the rest of the terms implicitly, that is:

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

Moreover in the ODE of the model 7, we evaluate w implicitly and we consider V_m at the previous temporal step t^k :

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

Therefore we obtain:

$$\begin{aligned}
& \chi_m C_m \begin{bmatrix} M & -M \\ -M & M \end{bmatrix} \begin{bmatrix} \frac{\phi_i^{k+1} - \phi_i^k}{\Delta t} \\ \frac{\phi_e^{k+1} - \phi_e^k}{\Delta t} \end{bmatrix} + \begin{bmatrix} A_i & 0 \\ 0 & A_e \end{bmatrix} \begin{bmatrix} \phi_i^{k+1} \\ \phi_e^{k+1} \end{bmatrix} + \\
& \begin{bmatrix} C(V_m^h) & -C(V_m^h) \\ -C(V_m^h) & C(V_m^h) \end{bmatrix} \begin{bmatrix} \phi_i^{k+1} \\ \phi_e^{k+1} \end{bmatrix} + \chi_m \begin{bmatrix} M & 0 \\ 0 & -M \end{bmatrix} \begin{bmatrix} w^{k+1} \\ w^{k+1} \end{bmatrix} = \begin{bmatrix} F_i^{k+1} \\ F_e^{k+1} \end{bmatrix} \quad (22) \\
& M \frac{w^{k+1} - w^k}{\Delta t} = \epsilon M (V_m^k - \gamma w^{k+1})
\end{aligned}$$

where $V_m^k = \phi_i^k - \phi_e^k$.

We can rewrite it as:

$$\left(\frac{\chi_m C_m}{\Delta t} \begin{bmatrix} M & -M \\ -M & M \end{bmatrix} + \begin{bmatrix} A_i & 0 \\ 0 & A_e \end{bmatrix} + \begin{bmatrix} C(V_m^k) & -C(V_m^k) \\ -C(V_m^k) & C(V_m^k) \end{bmatrix} \right) \begin{bmatrix} \phi_i^{k+1} \\ \phi_e^{k+1} \end{bmatrix} = \begin{bmatrix} F_i^{k+1} \\ F_e^{k+1} \end{bmatrix} - \chi_m \begin{bmatrix} M & 0 \\ 0 & -M \end{bmatrix} \begin{bmatrix} w^{k+1} \\ w^{k+1} \end{bmatrix} + \frac{\chi_m C_m}{\Delta t} \begin{bmatrix} M & 0 \\ 0 & -M \end{bmatrix} \begin{bmatrix} V_m^k \\ V_m^k \end{bmatrix} \quad (23)$$

$$\left(\frac{1}{\Delta t} + \epsilon \gamma \right) M w^{k+1} = \epsilon M V_m^k + \frac{M}{\Delta t} w^k \quad (24)$$

Defining:

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

We get the final system:

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

$$\begin{cases} \left(\frac{1}{\Delta t} + \epsilon \gamma \right) M w^{k+1} = \epsilon M V_m^k + \frac{M}{\Delta t} w^k \\ (B + C_{nl}(\Phi^k)) \Phi^{k+1} = r^{k+1} \end{cases} \quad (25)$$

4.1.1 Results

Same method for the time discretization used in (??) PLOT

4.2 Quasi implicit operator splitting

The main characteristic of an operator splitting is to divide the problem into two different systems with two different operators, such that $L(u) = L_1(u) + L_2(u)$, and, starting from u^n , we find \tilde{u}^{n+1} through the first system, then the solution u^{n+1} through the second one. In the Quasi implicit operator splitting we treat in a explicit way any term, except the V_m^n in the non linear

term.

Find \tilde{V}_m^{k+1} and w^{k+1} such that:

$$\begin{cases} \chi_m C_m M \frac{\tilde{V}_m^{k+1} - V_m^k}{\Delta t} + C(V_m^k) V_m^{k+1} + \chi_m M w^{k+1} = 0 \\ \frac{w^{k+1} - w^k}{\Delta t} = \epsilon(V_m^{k+1} - \gamma w^{k+1}) \end{cases} \quad (26)$$

Find V_m^{k+1} such that:

$$\begin{cases} \chi_m C_m M \frac{V_m^{k+1} - \tilde{V}_m^{k+1}}{\Delta t} + A_i \phi_i^{k+1} = F_i^{k+1} \\ -\chi_m C_m M \frac{V_m^{k+1} - \tilde{V}_m^{k+1}}{\Delta t} + A_e \phi_e^{k+1} = F_e^{k+1} \end{cases} \quad (27)$$

Put into a unique system:

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

Defining:

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

The equations in the system 28 can be written as:

1.

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

2.

$$\begin{aligned} \chi_m C_m M \frac{\phi_i^{k+1} - \phi_e^{k+1} - V_m^k}{\Delta t} + C(V_m^k)(\phi_i^{k+1} - \phi_e^{k+1}) + \chi_m M \left(\frac{w^k + \epsilon \Delta t (\phi_i^{k+1} - \phi_e^{k+1})}{1 + \epsilon \gamma \Delta t} \right) + \\ - A_e \phi_e^{k+1} = -F_e^{k+1} \\ \Rightarrow Q_k \phi_i^{k+1} - (Q_k + A_e) \phi_e^{k+1} = R_k - F_e^{k+1} \end{aligned}$$

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 $w^{k+1} \quad \forall k = 0, \dots, K-1$ such that:

$$\begin{cases} \left(\begin{bmatrix} Q_k & -Q_k \\ Q_k & -Q_k \end{bmatrix} + \begin{bmatrix} A_i & 0 \\ 0 & -A_e \end{bmatrix} \right) \begin{bmatrix} \phi_i^{k+1} \\ \phi_e^{k+1} \end{bmatrix} = \begin{bmatrix} R_k \\ R_k \end{bmatrix} + \begin{bmatrix} F_i^{k+1} \\ -F_e^{k+1} \end{bmatrix} \\ w^{k+1} = \frac{w^k + \epsilon \Delta t (\phi_i^{k+1} - \phi_e^{k+1})}{1 + \epsilon \gamma \Delta t} \end{cases} \quad (29)$$

4.2.1 Implementation

```
ZERO = sparse(11,11);

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(1:11)-u1(11+1:end);

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

    f0 = f1;
    Vm0 = u1(1:11) - u1(11+1:end);
    u0 = u1;
    w0 = w1;
end
```

4.2.2 Results

PLOT

4.3 Godunov operator splitting

Another type of operator splitting is the Godunov operator splitting where in this case the first system is written in an explicit way, meanwhile the second implicitly. Indeed:

Find \tilde{V}_m^{k+1} and w^{k+1} such that:

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

Find V_m^{k+1} such that:

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

Put in a unique system:

$$\begin{cases} \chi_m C_m M \frac{V_m^{k+1} - V_m^k}{\Delta t} + C(V_m^k) V_m^k + \chi_m M w^k + A_i \phi_i^{k+1} = F_i^{k+1} \\ \chi_m C_m M \frac{V_m^{k+1} - V_m^k}{\Delta t} + C(V_m^k) V_m^k + \chi_m M w^k - A_e \Phi_e^{k+1} = -F_e^{k+1} \\ w^{k+1} = (1 - \epsilon \gamma \Delta t) w^k + \epsilon \Delta t V_m^k \end{cases} \quad (30)$$

The equations in the system 30 can be written as:

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

In the end, find $\Phi^{k+1} = [\phi_i^{k+1} \phi_e^{k+1}]^T$ and $w^{k+1} \quad \forall k = 0, \dots, K-1$ such that:

$$\begin{cases} \left(\frac{\chi_m C_m}{\Delta t} \begin{bmatrix} M & -M \\ M & -M \end{bmatrix} + \begin{bmatrix} A_i & 0 \\ 0 & -A_e \end{bmatrix} \right) \begin{bmatrix} \phi_i^{k+1} \\ \phi_e^{k+1} \end{bmatrix} = \begin{bmatrix} F_i^{k+1} \\ -F_e^{k+1} \end{bmatrix} + \\ -\chi_m \begin{bmatrix} M & 0 \\ 0 & M \end{bmatrix} \begin{bmatrix} w^k \\ w^k \end{bmatrix} + \left(\frac{\chi_m C_m}{\Delta t} \begin{bmatrix} M & 0 \\ 0 & M \end{bmatrix} - \begin{bmatrix} C(V_m^k) & 0 \\ 0 & C(V_m^k) \end{bmatrix} \right) \begin{bmatrix} V_m^k \\ V_m^k \end{bmatrix} \\ w^{k+1} = (1 - \epsilon \gamma \Delta t) w^k + \epsilon \Delta t V_m^k \end{cases} \quad (31)$$

4.3.1 Implementation

```
ZERO = sparse(11,11);
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(1:11) - u1(11+1:end);
    u0 = u1;
    w0 = w1;
end
```

4.3.2 Results

PLOT

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] L. Marta and M. Perego. *Discontinuous Galerkin approximation of the bidomain system for cardiac electrophysiology*. Politecnico di Milano, 2021.
- [7] A. Quarteroni, A. Manzoni, and C. Vergara. “The cardiovascular system: Mathematical modelling, numerical algorithms and clinical applications”. in: *Acta Numerica* (2017), pp. 365–590.