NAPDE project outline

Matteo Calafà and Federica Botta

June 10, 2021

Contents

1 Monodomain

1.1 Analytical models

Monodomain model

$$\begin{cases} \chi_m C_m \frac{\partial V_m}{\partial t} - \nabla \cdot (\Sigma \nabla V_m) + \chi_m I_{ion}(V_m, w) = I^{ext} & \text{in } \Omega_{mus} \times (0,T] \\ \frac{\partial w}{\partial t} = g(V_m, w) & \text{in } \Omega_{mus} \times (0,T] \\ \Sigma \nabla V_m \cdot n = b & \text{on } \partial\Omega_{mus} \times (0,T] \end{cases} \tag{1}$$

where the unknowns are:

- $V_m = \Phi_i - \Phi_e$ (difference between internal and external potential)
- w ("*gating variable*")

and these constants are given : χ_m, C_m, Σ

FitzHugh-Nagumo model

$$\begin{gathered} I_{ion}(V_m, w) = kV_m(V_m - a)(V_m - 1) + w \\ g(V_m, w) = \epsilon(V_m - \gamma w) \end{gathered} \tag{2}$$

1.2 Semi-discretized numerical method

$$\left.\begin{aligned} V_{ij} &= \int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i \\ I^T_{i,j} &= \sum_{F \in F_h^I} \int_F \{\{\nabla\varphi_j\}\} \cdot [[\varphi_i]] \\ I_{i,j} &= \sum_{F \in F_h^I} \int_F [[\varphi_j]] \cdot \{\{\nabla\varphi_i\}\} \\ S_{i,j} &= \sum_{F \in F_h^I} \int_F \gamma [[\varphi_j]] \cdot [[\varphi_i]]] \end{aligned}\right\} \quad A = \Sigma(V - I^T - \delta I + S) \tag{3}$$

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

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

$$F_i = \int_\Omega f\varphi_i - \sum_{F \in F_h^B} \int_F b\varphi_i \tag{6}$$

Semi-discretized problem

$$\{\varphi_j\}_{j=1}^{N_h} \text{ base di } V_h^p = \{v_h \in L^2 : v_h|_K \in \mathbb{P}^{p_k}(K) \quad p_k \leq p \quad \forall K \in \tau_h\}$$
$$u_h(t) = \sum_{j=1}^{N_h} u_j(t)\varphi_j, \quad w_h(t) = \sum_{j=1}^{N_h} w_j(t)\varphi_j$$

$$\Rightarrow \quad \begin{array}{c} \boxed{\chi_m C_m M\dot{u} + Au + C(u_h)u + \chi_m M w = F} \\ \boxed{\dot{w} = \epsilon(u - \gamma w)} \end{array} \tag{7}$$

1.3 Totally-discretized numerical method ("$\boldsymbol{\theta - method}$")

Initial form ($\theta \in [0,1]$)

1.
$$\chi_m C_m M \frac{u^{k+1} - u^k}{\Delta t} + A\left(\theta u^{k+1} + (1-\theta)u^k\right) + C(u^k)\left(\theta u^{k+1} + (1-\theta)u^k\right) + \\ + \chi_m M w^{k+1} = \theta F^{k+1} + (1-\theta)F^k \tag{8}$$

2.
$$\frac{w^{k+1} - w^k}{\Delta t} = \epsilon(u^k - \gamma w^{k+1}) \tag{9}$$

Expanded form ($\theta \in [0,1]$)

1.
$$\left[\chi_m C_m M + \theta\Delta t A + \theta\Delta t C(u^k)\right] \boldsymbol{u^{k+1}} = \theta\Delta t F^{k+1} + (1-\theta)\Delta t F^k + \\ \left[\chi_m C_m M - (1-\theta)\Delta t A - (1-\theta)\Delta t C(u^k)\right] u^k - \chi_m \Delta t M w^{k+1} \tag{10}$$

2.
$$\left[1 + \epsilon\gamma\Delta t\right] \boldsymbol{w^{k+1}} = w^k + (\epsilon\Delta t)u^k \tag{11}$$

2 Bidomain

2.1 Analytical models

Bidomain model

$$\begin{cases} \chi_m C_m \frac{\partial V_m}{\partial t} - \nabla \cdot (\Sigma_i \nabla \phi_i) + \chi_m I_{ion}(V_m, w) = I_i^{ext} & \text{in } \Omega_{mus} \times (0,T] \\ -\chi_m C_m \frac{\partial V_m}{\partial t} - \nabla \cdot (\Sigma_e \nabla \phi_e) - \chi_m I_{ion}(V_m, w) = -I_e^{ext} & \text{in } \Omega_{mus} \times (0,T] \\ \frac{\partial w}{\partial t} = g(V_m, w) & \text{in } \Omega_{mus} \times (0,T] \\ \Sigma_i \nabla \phi_i \cdot n = b_i & \text{on } \partial\Omega_{mus} \times (0,T] \\ \Sigma_e \nabla \phi_e \cdot n = b_e & \text{on } \partial\Omega_{mus} \times (0,T] \end{cases} \tag{12}$$

where the unknowns are:

- $\phi_i, \phi_e \quad (V_m = \phi_i - \phi_e)$
- w ("*gating variable*")

and these constants are given: $\chi_m, C_m, \Sigma_i, \Sigma_e$

FitzHugh-Nagumo model

$$\begin{aligned} I_{ion}(V_m, w) &= kV_m(V_m - a)(V_m - 1) + w \\ g(V_m, w) &= \epsilon(V_m - \gamma w) \end{aligned} \tag{13}$$

2.2 Semi-discretized numerical methods

$$\left.\begin{aligned} V_{ij} &= \int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i \\ I^T_{i,j} &= \sum_{F \in F_h^I} \int_F \{\{\nabla\varphi_j\}\} \cdot [[\varphi_i]] \\ I_{i,j} &= \sum_{F \in F_h^I} \int_F [[\varphi_j]] \cdot \{\{\nabla\varphi_i\}\} \\ S_{i,j} &= \sum_{F \in F_h^I} \int_F \gamma [[\varphi_j]] \cdot [[\varphi_i]]] \end{aligned}\right\} \quad \begin{aligned} A &= (V - I^T - \theta I + S) \\ A_i &= \Sigma_i A \\ A_e &= \Sigma_e A \end{aligned} \tag{14}$$

$$M_{ij} = \sum_{K \in \tau_h} \int_K \varphi_j \varphi_i \tag{15}$$

$$C(u_h)_{ij} = \sum_{K \in \tau_h} \int_K \chi_m k(u_h - 1)(u_h - a)\varphi_j \varphi_i \tag{16}$$

$$\begin{aligned} F_{i,k} &= \int_\Omega I_i^{ext} \varphi_k - \sum_{F \in F_h^B} \int_F b_i \varphi_k \\ F_{e,k} &= -\int_\Omega I_e^{ext} \varphi_k - \sum_{F \in F_h^B} \int_F b_e \varphi_k \end{aligned} \tag{17}$$

Semi-discretized problem

$$\{\varphi_j\}_{j=1}^{N_h} \text{ base di } V_h^k = \{v_h \in L^2 : v_h|_{\mathcal{K}} \in \mathbb{P}^k(\mathcal{K}) \quad \forall \mathcal{K} \in \tau_h\}$$
$$\Phi_h(t) = \begin{bmatrix} \Phi_i^h(t) \\ \Phi_e^h(t) \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{N_h} \Phi_{i,j}(t)\varphi_j \\ \sum_{j=1}^{N_h} \Phi_{e,j}(t)\varphi_j \end{bmatrix}, \quad w_h(t) = \sum_{j=1}^{N_h} w_j(t)\varphi_j$$

$$\Rightarrow \quad \boxed{\begin{array}{c} \chi_m C_m \begin{bmatrix} M & -M \\ -M & M \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\Phi}}_{\boldsymbol{i}}^{\boldsymbol{h}}(\boldsymbol{t}) \\ \dot{\boldsymbol{\Phi}}_{\boldsymbol{e}}^{\boldsymbol{h}}(\boldsymbol{t}) \end{bmatrix} + \begin{bmatrix} A_i & 0 \\ 0 & A_e \end{bmatrix} \begin{bmatrix} \boldsymbol{\Phi}_{\boldsymbol{i}}^{\boldsymbol{h}}(\boldsymbol{t}) \\ \boldsymbol{\Phi}_{\boldsymbol{e}}^{\boldsymbol{h}}(\boldsymbol{t}) \end{bmatrix} + \\ \begin{bmatrix} C(V_m^h) & -C(V_m^h) \\ -C(V_m^h) & C(V_m^h) \end{bmatrix} \begin{bmatrix} \boldsymbol{\Phi}_{\boldsymbol{i}}^{\boldsymbol{h}}(\boldsymbol{t}) \\ \boldsymbol{\Phi}_{\boldsymbol{e}}^{\boldsymbol{h}}(\boldsymbol{t}) \end{bmatrix} + \chi_m \begin{bmatrix} M & 0 \\ 0 & -M \end{bmatrix} \begin{bmatrix} w_h(t) \\ w_h(t) \end{bmatrix} = \begin{bmatrix} F_i^h \\ F_e^h \end{bmatrix} \end{array}} \tag{18}$$

$$\boxed{\dot{w}_h(t) = \epsilon(V_m^h(t) - \gamma w_h(t))} \tag{19}$$

2.3 Totally-discretized numerical methods

2.3.1 Semi-implicit method

Initial form

$$\begin{array}{c} \chi_m C_m \begin{bmatrix} M & -M \\ -M & M \end{bmatrix} \begin{bmatrix} \frac{\Phi_i^{k+1} - \Phi_i^k}{\Delta t} \\ \frac{\Phi_e^{k+1} - \Phi_e^k}{\Delta t} \end{bmatrix} + \begin{bmatrix} A_i & 0 \\ 0 & A_e \end{bmatrix} \begin{bmatrix} \Phi_i^{k+1} \\ \Phi_e^{k+1} \end{bmatrix} + \\ \begin{bmatrix} C(V_m^k) & -C(V_m^k) \\ -C(V_m^k) & C(V_m^k) \end{bmatrix} \begin{bmatrix} \Phi_i^{k+1} \\ \Phi_e^{k+1} \end{bmatrix} + \chi_m \begin{bmatrix} M & 0 \\ 0 & -M \end{bmatrix} \begin{bmatrix} w^{k+1} \\ w^{k+1} \end{bmatrix} = \begin{bmatrix} F_i^{k+1} \\ F_e^{k+1} \end{bmatrix} \end{array} \tag{20}$$

$$\frac{w^{k+1} - w^k}{\Delta t} = \epsilon(V_m^k - \gamma w^{k+1}) \tag{21}$$

Expanded form

$$\begin{array}{c} \left(\frac{\chi_m C_m}{\Delta t} \begin{bmatrix} M & -M \\ -M & M \end{bmatrix} + \begin{bmatrix} A_i & 0 \\ 0 & A_e \end{bmatrix} + \begin{bmatrix} C(V_m^k) & -C(V_m^k) \\ -C(V_m^k) & C(V_m^k) \end{bmatrix} \right) \begin{bmatrix} \boldsymbol{\Phi}_{\boldsymbol{i}}^{\boldsymbol{k+1}} \\ \boldsymbol{\Phi}_{\boldsymbol{e}}^{\boldsymbol{k+1}} \end{bmatrix} = \\ \begin{bmatrix} F_i^{k+1} \\ F_e^{k+1} \end{bmatrix} - \chi_m \begin{bmatrix} M & 0 \\ 0 & -M \end{bmatrix} \begin{bmatrix} w^{k+1} \\ w^{k+1} \end{bmatrix} + \frac{\chi_m C_m}{\Delta t} \begin{bmatrix} M & -M \\ -M & M \end{bmatrix} \begin{bmatrix} \Phi_i^k \\ \Phi_e^k \end{bmatrix} \end{array} \tag{22}$$

$$(1 + \epsilon\gamma\Delta t)w^{k+1} = w^k + \epsilon\Delta t V_m^k \tag{23}$$

2.3.2 Quasi-implicit Operator Splitting

Initial form

I

$$\begin{aligned} \chi_m C_m M \frac{\tilde{V}_m^{n+1} - V_m^n}{\Delta t} + C(V_m^n) V_m^{n+1} + \chi_m M w^{n+1} = 0 \\ \frac{w^{n+1} - w^n}{\Delta t} = \epsilon (V_m^{n+1} - \gamma w^{n+1}) \end{aligned} \tag{24}$$

II

$$\begin{aligned} \chi_m C_m M \frac{V_m^{n+1} - \tilde{V}_m^{n+1}}{\Delta t} + A_i \Phi_i^{n+1} = F_i^{n+1} \\ -\chi_m C_m M \frac{V_m^{n+1} - \tilde{V}_m^{n+1}}{\Delta t} + A_e \Phi_e^{n+1} = F_e^{n+1} \end{aligned} \tag{25}$$

Expanded form

$$\begin{cases} \chi_m C_m M \frac{V_m^{n+1} - V_m^n}{\Delta t} + C(V_m^n) V_m^{n+1} + \chi_m M w^{n+1} + A_i \Phi_i^{n+1} = F_i^{n+1} \\ \chi_m C_m M \frac{V_m^{n+1} - V_m^n}{\Delta t} + C(V_m^n) V_m^{n+1} + \chi_m M w^{n+1} - A_e \Phi_e^{n+1} = -F_e^{n+1} \\ \frac{w^{n+1} - w^n}{\Delta t} = \epsilon (V_m^{n+1} - \gamma w^{n+1}) \end{cases} \tag{26}$$

$$\begin{aligned} &\bullet \quad Q_n := \frac{\chi_m C_m}{\Delta t} M + C(V_m^n) + \frac{\epsilon \chi_m \Delta t}{1 + \epsilon \gamma \Delta t} M \\ &\bullet \quad R_n := \frac{\chi_m C_m}{\Delta t} M V_m^n - \frac{\chi_m}{1 + \epsilon \gamma \Delta t} M w^n \end{aligned} \tag{27}$$

1.

$$\begin{aligned} \chi_m C_m M \frac{\Phi_i^{n+1} - \Phi_e^{n+1} - V_m^n}{\Delta t} + C(V_m^n)(\Phi_i^{n+1} - \Phi_e^{n+1}) + \\ \chi_m M \left(\frac{w^n + \epsilon \Delta t (\Phi_i^{n+1} - \Phi_e^{n+1})}{1 + \epsilon \gamma \Delta t} \right) + A_i \Phi_i^{n+1} = F_i^{n+1} \end{aligned} \tag{28}$$

$$\Rightarrow \quad (Q_n + A_i) \Phi_i^{n+1} - Q_n \Phi_e^{n+1} = R_n + F_i^{n+1}$$

2.

$$\begin{aligned} \chi_m C_m M \frac{\Phi_i^{n+1} - \Phi_e^{n+1} - V_m^n}{\Delta t} + \cdot C(V_m^n)(\Phi_i^{n+1} - \Phi_e^{n+1}) + \\ \chi_m M \left(\frac{w^n + \epsilon \Delta t (\Phi_i^{n+1} - \Phi_e^{n+1})}{1 + \epsilon \gamma \Delta t} \right) - A_e \Phi_e^{n+1} = -F_e^{n+1} \end{aligned} \tag{29}$$

$$\Rightarrow \quad Q_n \Phi_i^{n+1} - (Q_n + A_e) \Phi_e^{n+1} = R_n - F_e^{n+1}$$

3.

$$w^{n+1} = \frac{w^n + \epsilon \Delta t (\Phi_i^{n+1} - \Phi_e^{n+1})}{1 + \epsilon \gamma \Delta t} \tag{30}$$

$$\Rightarrow \begin{cases} \left(\begin{bmatrix} Q_n & -Q_n \\ Q_n & -Q_n \end{bmatrix} + \begin{bmatrix} A_i & 0 \\ 0 & -A_e \end{bmatrix} \right) \begin{bmatrix} \boldsymbol{\Phi_i^{n+1}} \\ \boldsymbol{\Phi_e^{n+1}} \end{bmatrix} = \begin{bmatrix} R_n \\ R_n \end{bmatrix} + \begin{bmatrix} F_i^{n+1} \\ -F_e^{n+1} \end{bmatrix} \\ Q_n = \frac{\chi_m C_m}{\Delta t} M + \cdot C(V_m^n) + \frac{\epsilon \chi_m \Delta t}{1+\epsilon\gamma\Delta t} M \\ R_n := \frac{\chi_m C_m}{\Delta t} M V_m^n - \frac{\chi_m}{1+\epsilon\gamma\Delta t} M w^n \\ w^{n+1} = \frac{w^n + \epsilon\Delta t(\Phi_i^{n+1} - \Phi_e^{n+1})}{1+\epsilon\gamma\Delta t} \end{cases} \tag{31}$$

2.3.3 Godunov Operator Splitting

Initial form

I

$$\begin{aligned} \chi_m C_m M \frac{\hat{V}_m^{n+1} - V_m^n}{\Delta t} + C(V_m^n) V_m^n + \chi_m M w^n = 0 \\ \frac{w^{n+1} - w^n}{\Delta t} = \epsilon(V_m^n - \gamma w^n) \end{aligned} \tag{32}$$

II

$$\begin{aligned} \chi_m C_m M \frac{V_m^{n+1} - \hat{V}_m^{n+1}}{\Delta t} + A_i \Phi_i^{n+1} = F_i^{n+1} \\ -\chi_m C_m M \frac{V_m^{n+1} - \hat{V}_m^{n+1}}{\Delta t} + A_e \Phi_e^{n+1} = F_e^{n+1} \end{aligned} \tag{33}$$

Expanded form

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

$$\Rightarrow \begin{cases} \left(\frac{\chi_m C_m}{\Delta t} M + A_i \right) \Phi_i^{n+1} - \frac{\chi_m C_m}{\Delta t} M \Phi_e^{n+1} = F_i^{n+1} - \chi_m M w^n + \left(\frac{\chi_m C_m}{\Delta t} M - C(V_m^n) \right) V_m^n \\ \frac{\chi_m C_m}{\Delta t} M \Phi_i^{n+1} - \left(\frac{\chi_m C_m}{\Delta t} M + A_e \right) \Phi_e^{n+1} = -F_e^{n+1} - \chi_m M w^n + \left(\frac{\chi_m C_m}{\Delta t} M - C(V_m^n) \right) V_m^n \\ w^{n+1} = (1 - \epsilon\gamma\Delta t) w^n + \epsilon \Delta t V_m^n \end{cases} \tag{34}$$

$$\Rightarrow \begin{cases} \left(\frac{\chi_m C_m}{\Delta t} \begin{bmatrix} M & -M \\ M & -M \end{bmatrix} + \begin{bmatrix} A_i & 0 \\ 0 & -A_e \end{bmatrix} \right) \begin{bmatrix} \boldsymbol{\Phi_i^{n+1}} \\ \boldsymbol{\Phi_e^{n+1}} \end{bmatrix} = \begin{bmatrix} F_i^{n+1} \\ -F_e^{n+1} \end{bmatrix} + \\ -\chi_m \begin{bmatrix} M & 0 \\ 0 & M \end{bmatrix} \begin{bmatrix} w^n \\ w^n \end{bmatrix} + \left(\frac{\chi_m C_m}{\Delta t} \begin{bmatrix} M & 0 \\ 0 & M \end{bmatrix} - \begin{bmatrix} C(V_m^n) & 0 \\ 0 & C(V_m^n) \end{bmatrix} \right) \begin{bmatrix} V_m^n \\ V_m^n \end{bmatrix} \\ w^{n+1} = (1 - \epsilon\gamma\Delta t) w^n + \epsilon \Delta t V_m^n \end{cases} \tag{35}$$

3 Dubiner basis

3.1 Basis functions

3.1.1 Mapping transformation

On the reference triangle

$$\hat{K} = \{(\xi, \eta) : \xi, \eta \geq 0, \xi + \eta \leq 1\} \tag{36}$$

we consider the transformation between the reference square and the reference triangle given by

$$\xi := \frac{(1+a)(1-b)}{4}, \eta := \frac{(1+b)}{2} \tag{37}$$

and the inverse transformation is

$$a := \frac{2\xi - 1 + \eta}{1-\eta} = \frac{2\xi}{1-\eta} - 1, b := 2\eta - 1 \tag{38}$$

3.1.2 Jacobian polynomials

Evaluation of the polynomial in $z \in \mathbb{R}^n$:

- $n = 0$

$$J_0^{\alpha,\beta}(z) = \overbrace{\begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}}^{\text{n times}} \tag{39}$$

- $n = 1$

$$J_1^{\alpha,\beta}(z) = \frac{1}{2}(\alpha - \beta + (\alpha + \beta + 2) \cdot z); \tag{40}$$

- $n \geq 2$

$$\begin{aligned} J_n^{\alpha,\beta}(z) = \sum_{k=2}^{n} \Big[& \frac{(2k+\alpha+\beta-1)(\alpha^2-\beta^2)}{2k(k+\alpha+\beta)(2k+\alpha+\beta-2)} + \\ & \frac{(2k+\alpha+\beta-2)(2k+\alpha+\beta-1)(2k+\alpha+\beta)}{2k(k+\alpha+\beta)(2k+\alpha+\beta-2)} J_{k-1}^{\alpha,\beta}(z) + \\ & - \frac{2(k+\alpha-1)(k+\beta-1)(2k+\alpha+\beta)}{2k(k+\alpha+\beta)(2k+\alpha+\beta-2)} J_{k-2}^{\alpha,\beta}(z) \Big] \end{aligned} \tag{41}$$

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

$$\int_{-1}^{1} (1-x)^\alpha (1+x)^\beta J_m^{\alpha,\beta} J_q^{\alpha,\beta}(x) dx = \frac{2}{2m+1} \delta_{mq} \tag{42}$$

3.1.3 Dubiner Basis

$$\begin{aligned}\phi_{ij}(\xi,\eta) :&= c_{ij}(1-b)^j J_i^{0,0}(a) J_j^{2i+1,0}(b) = \\ &= c_{ij}2^j(1-\eta)^j J_i^{0,0}(\frac{2\xi}{1-\eta}-1) J_j^{2i+1,0}(2\eta-1)\end{aligned} \tag{43}$$

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

$$c_{ij} := \sqrt{\frac{2(2i+1)(i+j+1)}{4^i}} \tag{44}$$

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

3.1.4 Gradient of Dubiner Basis

- $i = 0, j = 0$

$$\begin{aligned}\phi_{00}^{\xi}(\xi,\eta) &= 0 \\ \phi_{00}^{\eta}(\xi,\eta) &= 0\end{aligned} \tag{45}$$

- $i = 0, j \neq 0$

$$\begin{aligned}\phi_{0j}^{\xi} &= 0 \\ \phi_{0j}^{\eta} &= c_{0j}(j+2) J_{j-1}^{2,1}(b)\end{aligned} \tag{46}$$

- $i \neq 0, j = 0$

$$\begin{aligned}\phi_{i0}^{\xi}(\xi,\eta) &= c_{i0}2^i(1-\eta)^{i-1}(i+1) J_{i-1}^{1,1}(a) \\ \phi_{i0}^{\eta}(\xi,\eta) &= c_{i0}2^i(-i(1-\eta)^{i-1} J_i^{0,0}(a) + \xi(1-\eta)^{i-2}(i+1) J_{i-1}^{1,1}(a))\end{aligned} \tag{47}$$

- $i \neq 0, j \neq 0$

$$\begin{aligned}&\phi_{ij}^{\xi}(\xi,\eta) = c_{ij}2^i(1-\eta)^{i-1}(i+1) J_{i-1}^{1,1}(a) J_j^{2i+1,0}(b) \\ &\phi_{ij}^{\eta}(\xi,\eta) = c_{ij}2^i(-i(1-\eta)^{i-1} J_i^{0,0}(a) J_j^{2i+1,0}(b) + \xi(1-\eta)^{i-2}(i+1) J_{i-1}^{1,1}(a) J_j^{2i+1,0}(b) \\ &+ (1-\eta)^i(2i+j+2) J_i^{0,0}(a) J_{j-1}^{2i+2,1}(b))\end{aligned} \tag{48}$$

3.2 Transformation from FEM to Dubiner basis

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:

$$\psi_i(x_j) = \delta_{ij} \tag{49}$$

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 49, i.e. the coefficient $\hat{u}_i$ is nothing else but the evaluation of u_h on the i-th mesh point, then:

$$\hat{u}_i = \sum_{j=1}^{q} \tilde{u}_j \phi_j(x_i) \tag{50}$$

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

$$\tilde{u}_j = \int_{\mathcal{K}} u_h(x)\phi_j(x)\,dx = \int_{\mathcal{K}} \sum_{i=1}^{p} \hat{u}_i \psi_i(x)\phi_j(x)\,dx = \sum_{i=1}^{p} \Big(\int_{\mathcal{K}} \psi_i(x)\phi_j(x)\,dx \Big) \hat{u}_i \tag{51}$$

If the Dubiner functions are chosen as Galerkin basis, both the transformations are needed for the code implementation. Formula 50 is needed to plot and compute errors after the resolution of the system (otherwise solely Dubiner coefficients are useless). Formula 51 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, these are the coordinates of the evaluation points over the reference square:

- n=1

$$a = \begin{bmatrix} -1 & 1 & -1 \end{bmatrix}$$
$$b = \begin{bmatrix} -1 & -1 & 1 \end{bmatrix}$$

- n=2

$$a = \begin{bmatrix} 1 & 0 & 1 & 1 & -1 & -1 \end{bmatrix}$$
$$b = \begin{bmatrix} -1 & -1 & -1 & 0 & 1 & 0 \end{bmatrix}$$

- n=3

$$a = \begin{bmatrix} -1 & -0.5 & 0.5 & 1 & 1 & 1 & -1 & -1 & -1 & 0 \end{bmatrix}$$
$$b = \begin{bmatrix} -1 & -1 & -1 & -1 & -0.5 & 0.5 & 1 & 0.5 & -0.5 & -1/3 \end{bmatrix}$$

4 Few considerations about well-posedness

A proof of the well-posedness of the system coupled with the FitzHugh-Nagumo model is already available in an article of *Bourgault, Coudiere, Pierre*. However, the code was initially implemented with a wrong definition of the I_{ion}, due to an inverted sign. Using this formulation, we often found bad results that were probably caused by an ill-posedness of the problem. In this section we investigate a possible cause.
All the previous bidomain schemes treat the non-linear term in a semi-implicit way (see 18). It means that at time t^{n+1} we can solve a linear problem if data at time t^n are provided as parameters. From another perspective, at each time step t^{n+1}, we can fix the data of the previous time-step and solve a linear variational problem. In this case, the bilinear form at the left-hand side is the sum of some simpler bilinear forms: the standard mass L^2 inner product for the time derivative, the standard stiffness bilinear form and a pseudo-mass bilinear form for the non-linear term. With the inverted sign, this last bilinear form would be:

$$a_c(u,v) = -\int_\Omega \chi_m k(V_m^n - a)(V_m^n - 1)uv\,dx \tag{52}$$

Let us check that this bilinear form is coercive in $L^2(\Omega)$. That means to check that:

$$\exists\,\alpha > 0 : a_c(u,u) \geq \alpha ||u||^2_{L^2(\Omega)} \quad \forall u \in L^2(\Omega) \tag{53}$$

Then:

$$a_c(u,u) = \int_\Omega -\chi_m k(V_m^n - a)(V_m^n - 1)u^2\,dx \geq \inf_{x\in\Omega}[-\chi_m k(V_m^n - a)(V_m^n - 1)]||u||^2_{L^2} \tag{54}$$

Since all parameters are positive and $a \ll 1$, $\alpha = \inf_{x\in\Omega}[-\chi_m k(V_m^n - a)(V_m^n - 1)]$ is positive only if:

$$(V_m^n - 1)(V_m^n - a) < 0 \Leftrightarrow a < V_m^n < 1 \tag{55}$$

Finally, we can say that if the exact solution is between a and 1, a_c is coercive in $L^2(\Omega)$. Thanks to this result, the non-linear and the stiffness bilinear forms are together coercive in $H^1(\Omega)$ since the latter is coercive with respect to the L^2 semi-norm.

Obviously, this is only a sufficient condition for the H^1 total coercivity (remember that there is still the time derivative bilinear form that can contribute for the positiveness of $a(u,u)$). However, if this property fails, there may be bad consequences for the stability, especially in presence of physiological parameters. For the first simulations, for instance, we considered the exact solution:

$$V_m(x,t) = sin(2\pi x)sin(2\pi y)e^{-5t} \tag{56}$$

whose image is in [-1,1]. Setting some physiological parameters, we obtained some instabilities. Translating the exact solution, in order to replace the image into [0,1] (remember that a is a small parameter), we reached instead convergence:

$$V_m(x,t) = \frac{1}{2}sin(2\pi x)sin(2\pi y)e^{-5t} + \frac{1}{2} \tag{57}$$

5 Uniqueness of the ϕ solutions

From the bidomain system (12) a crucial fact can quickly be spotted: the internal and external potentials appear only through either their gradient or their difference V_m. It means that certainly, if ϕ_i, ϕ_e are solutions, then also $\phi_i + k$, $\phi_e + k$ are solutions $\forall k \in \mathbb{R}$. Fortunately, this problem could be easily solved by just imposing a further assumption on one of the two potentials. The most common impositions are:

$$\begin{aligned} \int_\Omega \phi_i \, dx = 0 \qquad & \text{(Null mean)} \\ \phi_i(x_j) = k \qquad & \text{(Fixed value in a certain mesh point)} \end{aligned} \tag{58}$$

However these choices would be tough to implement in a Dubiner context. In addition, it would be better to insert this condition directly in the system matrices avoiding to solve an ill-conditioned system. A simpler solution is certainly to fix one of the values of the solution vector, for instance the first. If $Au = b$ is the initial system to solve and $k \in \mathbb{R}$ is the chosen value for the coefficient, the problem can be modified as follows:

$$\tilde{A} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & a_{22} & \dots & a_{2N} \\ 0 & a_{32} & \dots & a_{3N} \\ \dots & \dots & \dots & \dots \\ 0 & a_{N2} & \dots & a_{NN} \end{bmatrix} \qquad \tilde{b} = \begin{bmatrix} k \\ b_2 - a_{21}k \\ b_3 - a_{31}k \\ \dots \\ b_N - a_{N1}k \end{bmatrix} \tag{59}$$

This procedure is certainly correct both for FEM and Dubiner basis. However, if one needs to test the convergence of the scheme, it would be better to provide the right value of k, i.e. the one related to the exact solution. But in this case, as already shown in section 3.2, the Galerkin coefficients have different meanings when using different basis. Summing up what have already been discussed, if $u \in H^1(\Omega)$ is the exact solution (sufficiently regular), then we can approximate it in the discretized space through the vector u_h defined as:

- FEM: the i-th coefficient is the value of u in the i-th mesh point
- Dubiner: the i-th coefficient is the L^2 scalar product of u with the i-th Dubiner polynomial

One can thus find in the code the evaluation of the exact solution in the first point of the mesh. For the latter case, instead, the code executes the integral over the first element of u_{ex} times the first basis function using the *Gauss-Legendre* quadrature nodes.

In conclusion, two benefits are achieved:

1. Even if the system was not singular, it was very ill-conditioned. This is because it discretized an ill-posed problem. With this slight modification, the condition number passed for instance from $\approx 10^{16-17}$ to $\approx 10^8$
2. If the coefficient is correctly set, one can observe and study the convergence of the computed solution towards the exact solution, without any kind of spurious vertical translation.