A Flexible Solver for the Monodomain Equation in MATLAB

Brodie A. J. Lawson

Abstract

A description of the code I have produced for the solution of the monodomain equation in two dimensions. This code is likely not a replacement for many advanced tools that are available, but for those familiar with MATLAB, it might serve well for preliminary simulations. A notable feature is the ability to specify unique diffusion tensors in each element, including zero tensors representing anatomical obstruction to excitation propagation. The solver uses an "intrinsic average" formulation, relevant for homogenisation techniques.

1 The Monodomain Equation

For now, the solver only considers the monodomain equation, a simplified yet widespread means of simulating electrical activity in the heart. The basic form of the monodomain equation is

$$C_m \frac{\partial V}{\partial t} = \frac{1}{\chi}\frac{\lambda}{1+\lambda}\nabla \cdot (\mathbf{\Sigma}\nabla V) + I_{\text{ion}} + I_{\text{stim}}, \tag{1}$$

with C_m the cell capacitance, χ the surface-to-volume ratio of the tissue, $\mathbf{\Sigma}$ the conductivity tensor and λ the ratio of intracellular and extracellular conductivities. I_{ion} and I_{stim} are the membrane current (flow of ions in and out of cell membranes) and stimulus current (injections of charge provided by the modeller), respectively.

This solver solves a slightly modified version of (1), that represents the "intrinsic average" membrane potential, a concept that considers only the membrane potential in the electrically active portions of space (that is, discluding regions occupied by things like non-conducting matrix proteins). The intrinsic average is introduced by defining the volume fraction of electrically active tissue, ϕ, and it can be shown that ϕ factors into equation (1), under certain assumptions, as

$$C_m \frac{\partial V}{\partial t} = \frac{1}{\chi}\frac{\lambda}{1+\lambda}\frac{1}{\phi}\nabla \cdot (\phi\mathbf{\Sigma}\nabla V) + I_{\text{ion}} + I_{\text{stim}}. \tag{2}$$

If ϕ is spatially constant, equation (2) reduces to (1), and so the solver may also be used to solve problems working with the traditional monodomain equation by making this choice.

2 Problem Specification

The finite volume method (FVM) solver used to timestep equation (2) for now works on a regular grid (see Section 3 for details), potentially to be extended in the future. A problem is specified by the user by defining the physical dimensions of this grid, Δx and Δy, in centimetres, as well as the value of $\mathbf{\Sigma}$ at each of its constituent volumes. This is specified by a series of matrices for the three components of the (symmetric) conduction tensor, σ_{xx}, σ_{xy} and σ_{yy}. This is different to typical monodomain solver implementations, which in two dimensions require the strengths of conduction in the longitudinal and transverse directions of myocardial fibres (σ_l, σ_t, respectively). For a fibre direction θ, the two approaches are linked by the relationship

$$\begin{pmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}^T \begin{pmatrix} \sigma_l & 0 \\ 0 & \sigma_t \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

and indeed this relationship can be used within a problem specification file to easily calculate the required values for the solver (see example). Importantly, if all components of the diffusion tensor are specified as zero at a location, then that location is rendered non-conductive, and no-flux boundary conditions are applied on its boundaries. The code adjusts to the introduction of these no-flux boundary conditions automatically. For example, node locations that are "buried" in non-conducting material will be removed from the list of nodes where the equations need to be solved.

In addition, the user must supply a function that defines which sites are or are not stimulus sites. As of current, only a single set of stimulus sites can be specified (that is to say, an S1-S2 stimulus protocol could not be implemented, although this should be a trivial extension). The values of the parameters χ, λ and C_m are all defined in the main code, as is the specification of I_{ion} and I_{stim}.

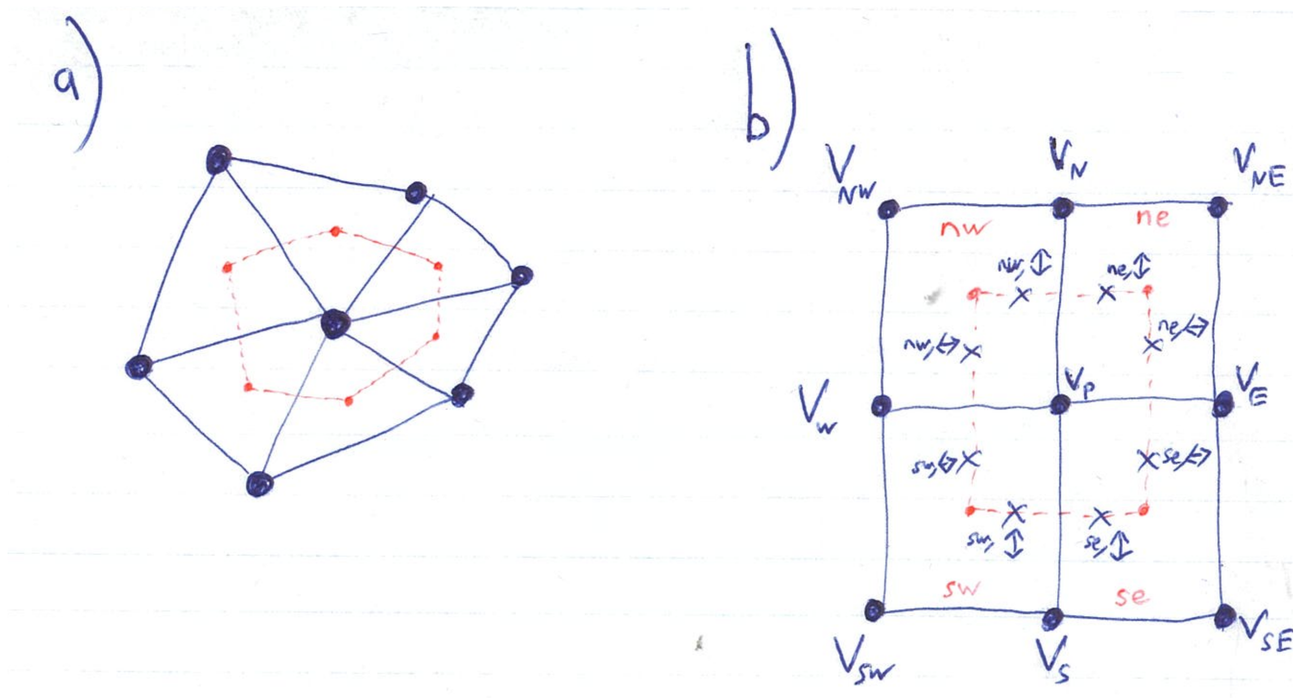

Figure 1: The vertex-centred finite volume approach, for **a)** an irregular grid and **b)** the regular grid used in this work. Red points indicate element centroids, red dotted lines are control volume faces. Blue crosses mark midpoints of control volume faces, where flux approximations are made. The image for **b)** also shows the notation used in this document.

3 Finite Volume Solution

3.1 Finite Volume Formulation

The solution of equation (2) is performed using a vertex-centred finite volume formulation. This approach places the nodes where information is known on the vertices of the mesh elements, and then control volumes are formed using the midpoints of the element faces and the mesh element centroids. This concept is visualised in Figure 1, which is also used to introduce the notation used here to describe the method.

The finite volume approach begins by integrating equation (2) over a control volume, here denoted Ω, and then dividing through by its volume, $\mathcal{V}$,

$$\frac{1}{\mathcal{V}}\int_\Omega C_m \frac{\partial V}{\partial t}\,d\Omega = \frac{1}{\mathcal{V}}\int_\Omega \frac{1}{\chi}\frac{\lambda}{1+\lambda}\frac{1}{\phi}\nabla\cdot\left(\phi\boldsymbol{\Sigma}\nabla V\right)d\Omega + \frac{1}{\mathcal{V}}\int_\Omega \left(I_{\text{ion}} + I_{\text{stim}}\right)d\Omega.$$

The order of differentiation and integration can be switched for the term on the left-hand side, at which point the integral divided by the volume becomes an average. Similarly, the source/sink term on the right is just the average of the two current terms over the volume. The diffusive term is treated using the divergence theorem, which allows it to be rewritten in terms of flux through the boundaries of Ω, here denoted S. Thus the equation becomes, after dividing through by capacitance C_m,

$$\frac{\partial \overline{V}}{\partial t} = -\frac{1}{\mathcal{V}}\int_S \frac{1}{\phi}\mathcal{J}\cdot\hat{\mathbf{n}}\,dS + \frac{1}{C_m}\left(\overline{I_{\text{ion}} + I_{\text{stim}}}\right), \tag{3}$$

with the flux $\mathcal{J}$ defined as

$$\mathcal{J} = -\frac{\lambda}{\chi C_m(1+\lambda)}\phi\boldsymbol{\Sigma}\nabla V.$$

The negatives are introduced here simply to ensure consistency with the traditional definition of flux.

The approximation applied to equation (3) is to replace the averages with the quantities calculated in terms of the centre of the control volume (which is the point P). This gives

$$\frac{\partial V_P}{\partial t} \approx -\frac{1}{\mathcal{V}}\int_S \frac{1}{\phi}\mathcal{J}\cdot\hat{\mathbf{n}}\,dS + \frac{1}{C_m}\Big(I_{\text{ion}}(V_P,\mathbf{s}_P) + I_{\text{stim}}|_P\Big),$$

where the dependence of the I_{ion} term on the membrane potential and the vector of state variables $\mathbf{s}$ is now made explicit. At this point all that remains is the approximation of the integral of the flux through the faces. The handling of this depends strongly on the regular grid (formulations for irregular grids also exist, of course). For a regular rectangular grid as considered here, there are eight faces associated with each control volume (see Figure 1). Flux through each is approximated separately, motivating the equation to be restated (after dropping the $\approx$ notation for cleanliness)

$$\frac{\partial V_P}{\partial t} = -\frac{1}{\mathcal{V}} \sum_{i=ne,nw,se,sw} \sum_{j=\updownarrow,\leftrightarrow} \int_{S_{i,j}} \frac{1}{\phi} \mathcal{J} \cdot \hat{\mathbf{n}}_{i,j} \, dS_{i,j} + \frac{1}{C_m}\Big(I_{\text{ion}}(V_P, \mathbf{s}_P) + I_{\text{stim}}|_P\Big).$$

This notation is perhaps a little arcane, but allows differentiation between which properties depend on only the element the face falls in, and those which depend on which face within the element is being considered (the face with an up/down normal or the face with a left/right normal).

3.2 Approximation of Flux Terms

The approximation that is made to allow progress is to approximate the flux integrals by assuming that the flux is constant across the face, and equal to the value at the face's midpoint. Denoting these approximations $\mathcal{J}_{i,j}$, and defining

$$\Delta_j = \begin{cases} \Delta x & j = \updownarrow \\ \Delta y & j = \leftrightarrow \end{cases},$$

our finite volume representation of the original equation now becomes

$$\frac{\partial V_P}{\partial t} = -\frac{1}{\phi_P}\frac{1}{\mathcal{V}} \sum_{i=ne,nw,se,sw} \sum_{j=\updownarrow,\leftrightarrow} \Delta_j \mathcal{J}_{i,j} \cdot \hat{\mathbf{n}}_{i,j} + I_{\text{ion}}(V_P, \mathbf{s}_P) + I_{\text{stim}}|_P \,. \tag{4}$$

The other thing that should be noted before continuing with an example of how the flux terms are approximated is that the quantity ϕ is associated with each *element*, not with each node. So, some sort of averaging will need to be used to calculate the value of ϕ_P, the value of ϕ associated with the node (discussed subsequently).

The quantity to be approximated is

$$\mathcal{J}_{i,j} \cdot \hat{\mathbf{n}}_{i,j} = -\frac{\lambda}{\chi C_m (1+\lambda)} \phi_i \Big(\mathbf{\Sigma}_i \nabla V_{i,j}\Big) \cdot \hat{\mathbf{n}}_{i,j}, \tag{5}$$

and the only thing unknown here is the value of $\nabla V_{i,j}$. To approximate this, we use bilinear interpolation, generating approximate values for V and ∇V across the whole element, using the values of V at the nodepoints in its four corners. As an example, we consider the flux through the faces $\mathcal{J}_{ne,\updownarrow}$ and $\mathcal{J}_{ne,\leftrightarrow}$, which will be approximated in terms of the values at the set of nodes at the vertices of the "*ne*" element, (V_P, V_E, V_N, V_{NE}).

By defining the bottom-left corner of the example element as $(0,0)$, the bilinear interpolation approximation for V is defined

$$V(x,y) = \frac{1}{\Delta x \Delta y}\Big((\Delta x - x)(\Delta y - y)V_P + x(\Delta y - y)V_E + (\Delta x - x)yV_N + xyV_{NE}\Big),$$

resulting in an expression for ∇V,

$$\nabla V(x,y) = \frac{1}{\Delta x \Delta y}\Bigg(\Big((\Delta y - y)(V_E - V_P) + y(V_{NE} - V_N)\Big)\mathbf{i} + \Big((\Delta x - x)(V_N - V_P) + x(V_{NE} - V_E)\Big)\mathbf{j}\Bigg).$$

The values of $\nabla V_{i,j}$ then become

$$\begin{aligned}
\nabla V_{ne,\updownarrow} &= \nabla V\left(\frac{\Delta x}{4}, \frac{\Delta y}{2}\right) = \frac{1}{\Delta x}\left(\frac{1}{2}V_E - \frac{1}{2}V_P + \frac{1}{2}V_{NE} - \frac{1}{2}V_N\right)\mathbf{i} + \frac{1}{\Delta y}\left(\frac{3}{4}V_N - \frac{3}{4}V_P + \frac{1}{4}V_{NE} - \frac{1}{4}V_E\right)\mathbf{j} \\
\nabla V_{ne,\leftrightarrow} &= \nabla V\left(\frac{\Delta x}{2}, \frac{\Delta y}{4}\right) = \frac{1}{\Delta x}\left(\frac{3}{4}V_E - \frac{3}{4}V_P + \frac{1}{4}V_{NE} - \frac{1}{4}V_N\right)\mathbf{i} + \frac{1}{\Delta y}\left(\frac{1}{2}V_N - \frac{1}{2}V_P + \frac{1}{2}V_{NE} - \frac{1}{2}V_E\right)\mathbf{j}.
\end{aligned}$$

The conduction tensor will potentially have non-diagonal elements, and thus both components of $\nabla V_{i,j}$ can feature in the flux calculations (5). Specifically, to complete this example,

$$\begin{aligned}
\mathcal{J}_{ne,\updownarrow} \cdot \hat{\mathbf{n}}_{ne,\updownarrow} = -\frac{\lambda}{\chi C_m(1+\lambda)}\phi_{ne}\Bigg(&\frac{1}{\Delta x}(\sigma_{xy})_{ne}\left(\frac{1}{2}V_E - \frac{1}{2}V_P + \frac{1}{2}V_{NE} - \frac{1}{2}V_N\right) \\
&+ \frac{1}{\Delta y}(\sigma_{yy})_{ne}\left(\frac{3}{4}V_N - \frac{3}{4}V_P + \frac{1}{4}V_{NE} - \frac{1}{4}V_E\right)\Bigg),
\end{aligned}$$

and similarly for $\mathcal{J}_{ne,\leftrightarrow}$ and the fluxes associated with the faces falling in the other surrounding elements.

In the case of an element where no surrounding elements are non-conductive, the full approximation of equation (4) becomes

$$\frac{\partial V_P}{\partial t} = \frac{1}{\phi_P} f_{\text{diffusive}}(V_P, V_E, V_W, V_N, V_S, V_{NE}, V_{NW}, V_{SE}, V_{SW}) + f_{\text{reaction}}(V_P, \mathbf{s}), \tag{6}$$

where

$$f_{\text{reaction}} = \frac{1}{C_m}\Big(I_{\text{ion}}(V_P, \mathbf{s}_P) + I_{\text{stim}}\big|_P\Big),$$

and

$$\begin{aligned}
f_{\text{diffusive}} = \frac{\lambda}{\chi C_m(1+\lambda)}\Bigg(\bigg[&-\frac{3}{8\Delta x^2}\Big((\phi\sigma_{xx})_{ne} + (\phi\sigma_{xx})_{nw} + (\phi\sigma_{xx})_{se} + (\phi\sigma_{xx})_{sw}\Big) \\
&+ \frac{1}{2\Delta x\Delta y}\Big(-(\phi\sigma_{xy})_{ne} + (\phi\sigma_{xy})_{nw} + (\phi\sigma_{xy})_{se} - (\phi\sigma_{xy})_{sw}\Big) \\
&- \frac{3}{8\Delta y^2}\Big((\phi\sigma_{yy})_{ne} + (\phi\sigma_{yy})_{nw} + (\phi\sigma_{yy})_{se} + (\phi\sigma_{yy})_{sw}\Big)\bigg]V_P \\
+\bigg[&\frac{1}{8\Delta x^2}(\phi\sigma_{xx})_{ne} + \frac{1}{2\Delta x\Delta y}(\phi\sigma_{xy})_{ne} + \frac{1}{8\Delta y^2}(\phi\sigma_{yy})_{ne}\bigg]V_{NE} \\
+\bigg[&\frac{1}{8\Delta x^2}(\phi\sigma_{xx})_{nw} - \frac{1}{2\Delta x\Delta y}(\phi\sigma_{xy})_{nw} + \frac{1}{8\Delta y^2}(\phi\sigma_{yy})_{nw}\bigg]V_{NW} \\
+\bigg[&\frac{1}{8\Delta x^2}(\phi\sigma_{xx})_{se} - \frac{1}{2\Delta x\Delta y}(\phi\sigma_{xy})_{se} + \frac{1}{8\Delta y^2}(\phi\sigma_{yy})_{se}\bigg]V_{SE} \\
+\bigg[&\frac{1}{8\Delta x^2}(\phi\sigma_{xx})_{sw} + \frac{1}{2\Delta x\Delta y}(\phi\sigma_{xy})_{sw} + \frac{1}{8\Delta y^2}(\phi\sigma_{yy})_{sw}\bigg]V_{SW} \\
+\bigg[&\frac{3}{8\Delta x^2}\Big((\phi\sigma_{xx})_{ne} + (\phi\sigma_{xx})_{se}\Big) - \frac{1}{8\Delta y^2}\Big((\phi\sigma_{yy})_{ne} + (\phi\sigma_{yy})_{se}\Big)\bigg]V_E \\
+\bigg[&\frac{3}{8\Delta x^2}\Big((\phi\sigma_{xx})_{nw} + (\phi\sigma_{xx})_{sw}\Big) - \frac{1}{8\Delta y^2}\Big((\phi\sigma_{yy})_{nw} + (\phi\sigma_{yy})_{sw}\Big)\bigg]V_W \\
+\bigg[&-\frac{1}{8\Delta x^2}\Big((\phi\sigma_{xx})_{ne} + (\phi\sigma_{xx})_{nw}\Big) + \frac{3}{8\Delta y^2}\Big((\phi\sigma_{yy})_{ne} + (\phi\sigma_{yy})_{nw}\Big)\bigg]V_N \\
+\bigg[&-\frac{1}{8\Delta x^2}\Big((\phi\sigma_{xx})_{se} + (\phi\sigma_{xx})_{sw}\Big) + \frac{3}{8\Delta y^2}\Big((\phi\sigma_{yy})_{se} + (\phi\sigma_{yy})_{sw}\Big)\bigg]V_S\Bigg).
\end{aligned}$$

As terrible as the approximation to the diffusive term looks, when $\phi = 1$ everywhere (no occlusions inside elements) and the diffusion tensor is constant across different elements, it reduces to

$$\begin{aligned}
f_{\text{diffusive}} = \frac{\lambda}{\chi C_m(1+\lambda)}\Bigg(&\bigg[-\frac{3}{2\Delta x^2}\sigma_{xx} - \frac{3}{2\Delta y^2}\sigma_{yy}\bigg]V_P \\
&+\bigg[\frac{1}{8\Delta x^2}\sigma_{xx} + \frac{1}{2\Delta x\Delta y}\sigma_{xy} + \frac{1}{8\Delta y^2}\sigma_{yy}\bigg]V_{NE} \\
&+\bigg[\frac{1}{8\Delta x^2}\sigma_{xx} - \frac{1}{2\Delta x\Delta y}\sigma_{xy} + \frac{1}{8\Delta y^2}\sigma_{yy}\bigg]V_{NW} \\
&+\bigg[\frac{1}{8\Delta x^2}\sigma_{xx} - \frac{1}{2\Delta x\Delta y}\sigma_{xy} + \frac{1}{8\Delta y^2}\sigma_{yy}\bigg]V_{SE} \\
&+\bigg[\frac{1}{8\Delta x^2}\sigma_{xx} + \frac{1}{2\Delta x\Delta y}\sigma_{xy} + \frac{1}{8\Delta y^2}\sigma_{yy}\bigg]V_{SW} \\
&+\bigg[\frac{3}{4\Delta x^2}\sigma_{xx} - \frac{1}{4\Delta y^2}\sigma_{yy}\bigg]\Big(V_E + V_W\Big) \\
&+\bigg[-\frac{1}{4\Delta x^2}\sigma_{xx} + \frac{3}{4\Delta y^2}\sigma_{yy}\bigg]\Big(V_N + V_S\Big)\Bigg),
\end{aligned}$$

which perhaps looks more like a reasonable discretisation for a "Laplacian with diagonal terms" in the nine-point stencil framework used by the vertex-centred approach. The coefficients in front of all other nodes except for V_P add up to the coefficients in front of V_P, and the "cross-term" coefficients add to zero. However this is my first time taking on the vertex-centred case so if this looks funny please let me know!

In cases where elements are occupied with occlusive material (such as fibrosis), the flux contributions associated with those elements are simply removed. Technically the control volume can now be thought of as running along the edges of the occluded elements, where a no-flux boundary condition is applied. In the MATLAB implementation, flux values associated with these elements are simply set to zero.

3.3 Approximation of Volume Fraction at Nodes

Equation (6) is now fully defined (excluding for the moment the fact that the I_{ion} term has not been discussed at all), except for the need to generate approximated values of ϕ_P. There are actually two approaches to this that seem reasonable. In this document I briefly describe both, before discussing what the implementation actually does.

The harmonic average is what is used to take the average of rates, and as the $1/\phi_P$ term sits in front of the diffusive contribution to the rate of change of membrane potential, it is natural to think of it as a rate. However, the value to be averaged is ϕ_P, and this is the reciprocal of the rate. The harmonic average of reciprocals is actually the reciprocal of the arithmetic (traditional) average, and so that is what is used in this work.

The question arises in the handling of nodes that are partially surrounded by non-conducting elements. To most easily illustrate the concepts here, consider the case where $\phi = 1$ everywhere, except for occluded elements where $\phi = 0$. This is equivalent to assuming that occlusions align perfectly with the grid and completely occupy the grid sites in which they fall (or put another way, some "pixels" are fully blocked, while the rest are as normal). The two approaches depend on whether the occluded elements are included in calculations for the average value of ϕ, or not. If they are, then the values for ϕ_P at nodes will correctly reflect the intuition that the volume fraction in that area is $\phi_P < 1$ because some of the surrounds are obstructed. However, it also destroys the (desirable?) property that the $\phi = 1$ case corresponds to the traditional monodomain equation, equation (1). On the other hand, if occluded elements are completely removed from the averaging, then if all non-occluded elements have $\phi = 1$, the averaged equation (2) will also consistently have $\phi_P = 1$ and be equivalent to equation (1).

The choice that must be made is between keeping consistency with the traditional equation, or fully committing to the concept of "intrinsic average" that equation (2) is derived using. Smaller values of ϕ will result in larger contributions of diffusive flux in those regions, and thus equates to the build-up of potential in these regions. Although this makes some physical sense, it is also disagreeable in that when not performing any homogenisation, the traditional monodomain equation (1) solved on the fine scale with no-flux conditions applied to the occluded regions is surely the ground truth, and surely it is desirable that our implementation of the averaged equation match it as best as possible. There is a subtle point that the V in the "ground truth" case (1) is different to the intrinsic average V case (2), and indeed one can be converted back to the other by using the volume fraction. However in the practise of solving the equations, I have to imagine that allowing extra potential to build up on boundaries will also change how well that charge spreads to neighbouring elements (consider for example invasion of a wavefront of excitation through an isthmus in scar tissue).

On the other hand, to continue talking this point into the ground, as soon as one moves away from the $\phi = 1$ case, representing the situation where there are small occlusions *within* elements that are not resolved by the problem and instead being handled by the switch to intrinsic average formulation (2) and appropriate modifications to the diffusion tensor in each element, it seems very arbitrary to average over these occlusions but not over occlusions represented by completely obstructed material. So I really am in two minds regarding this.

For these reasons, a switch has been included in the code, that controls which elements are or are not included in averaging for the calculation of ϕ_P values. The user can select for no obstructions to contribute to averages, for only "internal" obstructions to contribute (i.e. the obstructions implied by no-flux boundaries on the outside of the problem domain don't count), or for all obstructions to contribute. I still cannot really decide what is most appropriate here, so any thoughts on this matter are appreciated.

3.4 Timestepping

For now, timestepping of equation (6) is handled using a very simple operator split approach. This approach considers the contributions of $f_{\text{diffusive}}$ and f_{reaction} separately, allowing for the different natures of the separate terms to be taken full advantage of. Specifically, the form of $f_{\text{diffusive}}$ is linear, but expressed in terms of the nine-point stencil surrounding each nodepoint. This invites a matrix approach that easily permits implicit timestepping. On the other hand, the f_{reaction} term consists of a set of strongly nonlinear ordinary differential equations, for which implicit approaches are far more complex, but good results can be obtained with Rush–Larsen timestepping, a hybrid explicit–exponential integration approach described subsequently.

We first re-write equation (6) in terms of all nodes at once, simply by defining a vector $\mathbf{V}$ of membrane potential values. The vector of state values will also be different at each nodepoint, so $\mathbf{s}$ now becomes a matrix of the state variables at each nodepoint, $\mathcal{S}$. The equation thus becomes

$$\frac{\partial \mathbf{V}}{\partial t} = \frac{1}{\phi_P}\mathbf{f}_{\text{diffusive}}(\mathbf{V}) + \mathbf{f}_{\text{reaction}}(\mathbf{V}, \mathcal{S}).$$

Applying the standard first-order finite difference approximation to the left-hand side of this equation,

$$\frac{\mathbf{V}^{\text{new}} - \mathbf{V}^{\text{old}}}{\Delta t} = \frac{1}{\phi_P}\mathbf{f}_{\text{diffusive}}(\mathbf{V}) + \mathbf{f}_{\text{diffusive}}(\mathbf{V}, \mathcal{S}).$$

The operator split converts this equation into two separate updates applied sequentially, the first of which derives from the diffusive part evaluated at the "new" values (implicit timestepping)

$$\mathbf{V}^{\text{new}} = \mathbf{V}^{\text{old}} + \frac{\Delta t}{\phi_P}\mathbf{f}_{\text{diffusive}}(\mathbf{V}^{\text{new}}).$$

Note that there is some abuse of notation here, because ϕ_P can be different at each nodepoint, and thus the $1/\phi_P$ is actually a vector that element-wise multiplies (Hadamard product) with the vector of values composing $\mathbf{f}_{\text{diffusive}}$. Given that each $f_{\text{diffusive}}$ is a linear combination of V values at different nodepoints, this linear system can be expressed

$$\left(I - \frac{\Delta t}{\phi_P}\mathbf{A}\right)\mathbf{V}^{\text{new}} = \mathbf{V}^{\text{old}}$$
$$\mathbf{V}^{\text{new}} = \left(I - \frac{\Delta t}{\phi_P}\mathbf{A}\right)^{-1}\mathbf{V}^{\text{old}},$$

with the matrix $\mathbf{A}$ being sparse. This is solved in the code simply using MATLAB's backslash command. Note that nodepoints that are completely buried in occlusion are identified and removed. These correspond to rows of all zeros in the matrix $\mathbf{A}$ and so do not change in the update step. Identification of these nodes also saves calculation in the reaction term.

The second update is the reaction term update. This takes the form

$$\mathbf{V}^{\text{new}} = \mathbf{V}^{\text{old}} + \Delta t\,\mathbf{f}_{\text{reaction}}(\mathbf{V}^{\text{old}}, \mathcal{S}).$$

This appears to indicate explicit timestepping, and indeed this is what is used to update the value of $\mathbf{V}$ and some of the state variables. A subset of state variables, the gating variables, have a set form that enables "exact" integration, under the assumption that $\mathbf{V}$ is being timestepped explicitly. Specifically, the gating variables g are updated by the ordinary differential equation

$$\frac{dg}{dt} = \frac{g_\infty(V) - g}{\tau_g(V)}.$$

When the value of V is considered fixed, then this equation can be solved analytically to give an "exact" update formula,

$$g^{\text{new}} = g_\infty + \left(g^{\text{old}} - g_\infty\right) e^{-\Delta t/\tau_m}.$$

The Rush–Larsen updates used here first update all gating variables using this formula (using $\mathbf{V}^{\text{old}}$), then update the remaining state variables and calculate I_{ion} in order to update $\mathbf{V}$.