1 System and Gradient Derivation

The system of equations we solve is a typical system of reaction-diffusion PDEs:

\[
c_t = D_c c_{xx} - k_b^+ c(B_{tot} - b) + k_b^- b + \alpha \delta(x - x_u) \delta(t - t_u) \\
b_t = D_b b_{xx} + k_b^+ c(B_{tot} - b) - k_b^- b \\
c_x(0, t) = c_x(\ell, t) = 0 \\
b_x(0, t) = b_x(\ell, t) = 0
\]

where \( B_{tot} \) is the total concentration of buffered calcium injected into the cell, \( k_b^+ \), \( k_b^- \) are binding and unbinding rate constants, \( D_b, D_c \) are diffusion constants for buffered and unbuffered calcium, respectively. Also, \( \alpha \) is a dimensionless scaling term for the source term for calcium, coming from the precise uncaging of calcium.

For this first stage, we don’t concern ourselves with the calcium diffusion equation, just the buffered calcium PDE. We are given some experimental data, which are measurements of buffered calcium at \( N \) points in space across time. Our goal is to minimize the misfit function, given some experimental data \( \{b^*(x_j, t)\}_{j=1}^N \), over all possible calcium functions \( c(x, t) \):

\[
\min_c \Phi(c) = \min_c \left\{ \frac{1}{2} \sum_{j=1}^N \int_0^T (b^*(x_j, t) - b(x_j, t; c))^2 dt \right\}
\]

We define the Lagrangian which introduces an adjoint variable \( B(x, t) \):

\[
L(c, b, B) = \frac{1}{2} \sum_{j=1}^N \int_0^T (b^*(x_j, t) - b(x_j, t))^2 dt \\
+ \int_0^T \int_0^\ell (b_t - D_b b_{xx} - k_b^+ c(B_{tot} - b) + k_b^- b) Bdx dt
\]
It is important to take note that

$$\Phi(c) = \min_b \max_B L(c, b, B)$$

(4)

This allows us to write the gradient of $\Phi$ in some direction $\tilde{c}$:

$$\langle \partial \Phi(c), \tilde{c} \rangle = \langle \partial_c L(c, b, B), \tilde{c} \rangle = - \int_0^T \int_0^\ell b^+ (B_{tot} - b) B \tilde{c} dx dt$$

if $L$ is at a critical point,

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial B} = 0$$

Requiring that $\frac{\partial L}{\partial B} = 0$ is equivalent to (1). Seeing the implications of $\frac{\partial L}{\partial b} = 0$ requires a bit more work. We rewrite this as $\langle \partial_b L(c, b, B), \tilde{b} \rangle = 0$ for all possible $\tilde{b}$:

$$0 = \langle \partial_b L(c, b, B), \tilde{b} \rangle = \lim_{\varepsilon \to 0} \frac{L(c, b - \varepsilon \tilde{b}, B) - L(c, b, B)}{\varepsilon}$$

$$= \sum_{j=1}^N \int_0^T - (\tilde{b}^x(x_j, t) - b(x_j, t)) \tilde{b}(x_j, t) dt$$

$$+ \int_0^T \int_0^\ell (\tilde{b} - D_b \tilde{b}_x + k_b^+ \tilde{b} + k_b^- \tilde{b}) B dx dt$$

(5)

Now after enforcing that $\tilde{b}_x(0, t) = \tilde{b}_x(\ell, t) = \tilde{b}(x, 0) = 0$ we continue with integration by parts in an effort to rearrange this expression to a more convenient form:

$$\int_0^T \int_0^\ell \tilde{b}_1 B dx dt = \int_0^\ell \left( \tilde{b} B \right)_{t=0}^{t=T} dx - \int_0^T \int_0^\ell \tilde{b} B_t dx dt$$

$$= \int_0^\ell \tilde{b}(x, T) B(x, T) dx - \int_0^T \int_0^\ell \tilde{b} B_t dx dt$$

(7)

$$- D_b \int_0^T \int_0^\ell \tilde{b}_{xx} B dx dt = - D_b \left( \int_0^T \left[ \tilde{b}_x B \right]_{x=0}^{x=\ell} dt - \int_0^T \left[ \tilde{b} B_x \right]_{x=0}^{x=\ell} dt + \int_0^T \int_0^\ell \tilde{b} B_{xx} dx dt \right)$$

$$= D_b \left( \int_0^T \left[ \tilde{b} B_x \right]_{x=0}^{x=\ell} dt + \int_0^T \int_0^\ell \tilde{b} B_{xx} dx dt \right)$$

(8)

2
Using these allows us to obtain our desired form:

\[ 0 \] = \left< \partial_b L(c, b, B), \tilde{b} \right> \\
= \int_0^T -D_b \hat{b}(0, t) B_x(0, t) dt + \int_0^T D_b \hat{b}(\ell, t) B_x(\ell, t) dt + \int_0^T \hat{b}(x, T) B(x, T) dx \\
+ \int_0^T \left[ (B_t - D_b B_{xx} + k_b^+ c B + k_b^- B + \sum_{j=1}^N \delta(x - x_j)(b - b^*) \right] \tilde{b} dx dt \tag{10} \]

For this to hold, we see our adjoint variable must solve the following adjoint PDE:

\[- B_t - D_b B_{xx} + (k_b^+ c + k_b^-) B + \sum_{j=1}^N \delta(x - x_j)(b - b^*) = 0 \]

\[ B(x, T) = 0 \]

\[ B_x(0, t) = B_x(\ell, t) = 0 \tag{11} \]

So given some \( c \), we then solve both (1) and (11) to obtain a \( b(x, t) \) and \( B(x, t) \) over \([0, \ell] \times [0, T]\), and with this we can then obtain the gradient of \( \Phi \) in a direction \( \tilde{c} \) using (5). With this gradient in hand, we can dramatically increase the efficiency of our optimization algorithm, which will probably be Matlab’s \texttt{fminunc}.  

## 2 Obtaining \( b(x, t) \) and \( B(x, t) \)

Now concerning the implementation of these ideas, we turn to finite elements for handling space, and forward/backward Euler for time. For our original equation:

\[ D_b b_{xx} = b_t - k_b^+ c B_{tot} + (k_b^+ + k_b^-) b \]

\[ \int_0^\ell D_b b_{xx} vd\tau = \int_0^\ell (b_t - k_b^+ c B_{tot} + (k_b^+ - k_b^-) b) v d\tau \]

\[ - \int_0^\ell D_b b_x v_x d\tau = \int_0^\ell (b_t - k_b^+ c B_{tot} + (k_b^+ - k_b^-) b) v d\tau \]

Now let \( b = \sum_{k=1}^{N_x} b_k(t) \phi_k(x), \ c = \sum_{k=1}^{N_x} c_k(t) \phi_k(x), \) and after letting \( v = \phi_n(x) \) (where the \( \phi_k \)'s are the typical hat functions on our \( x \) grid) for \( n = 1, 2, \ldots, N_x \) we get the system:
\[-D_b \sum_{k=1}^{N_x} \int_0^\ell b_k(t) \phi'_k \phi'_n \, dx = \int_0^\ell \sum_{k=1}^{N_x} \partial_t b_k \phi_k \phi_n \, dx\]

\[+ \int_0^T \sum_{k=1}^{N_x} \sum_{j=1}^{N_x} k_j b_j \phi_k \phi_j \phi_n \, dx + \int_0^\ell \sum_{k=1}^{N_x} [k_b^{-} b_k - k_b^{+} c_k] \phi_k \phi_n \, dx \]

for each \( n = 1, 2, \ldots, N_x \).

Define \( d_k = x_{k+1} - x_k \) for \( k = 1, 2, \ldots, N_x \). Now we define the following square matrices:

\[
M = \begin{bmatrix}
\frac{d_1}{d_1 + d_2} & \frac{d_1}{d_1 + d_2} & \cdots & \frac{d_1}{d_1 + d_M} \\
\frac{d_2}{d_1 + d_2} & \frac{d_2}{d_1 + d_2} & \cdots & \frac{d_2}{d_1 + d_M} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{d_M}{d_1 + d_2} & \frac{d_M}{d_1 + d_2} & \cdots & \frac{d_M}{d_1 + d_M}
\end{bmatrix}
\]

\[K = \begin{bmatrix}
\frac{1}{d_1} & -\frac{1}{d_1} & 0 & \cdots & 0 \\
\frac{1}{d_1} + \frac{1}{d_2} & -\frac{1}{d_2} & 0 & \cdots & 0 \\
0 & \frac{1}{d_2} + \frac{1}{d_3} & -\frac{1}{d_3} & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & -\frac{1}{d_{N_x-1}}
\end{bmatrix}
\]

For the next matrix, call \( M = N_x - 1 \) to conserve space:

\[
L(c(:, t)) = \begin{bmatrix}
\frac{c_1 d_1 + c_2 d_2}{12} & \frac{d_1 (c_1 + c_2)}{12} & \cdots & \frac{d_2 (c_2 + c_3)}{12} \\
\frac{c_2 (d_1 + d_2)}{4} & \frac{c_2 d_1 + c_3 d_2}{12} & \cdots & \frac{c_3 d_2}{12} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{d_N (c_{N_x-2} + c_{N_x})}{4} & \frac{c_N (d_{N_x-2} + d_{N_x})}{12} & \cdots & \frac{c_N d_{N_x}}{12}
\end{bmatrix}
\]

Notice that \( L \) changes with time, so care must be taken in the code to update \( L \) at every timestep. Using these matrices, (15) now becomes

\[-D_b K b = M \frac{db}{dt} + k_b^{+} L(c(:, t)) b + k_b^{-} M b - k_b^{-} B_{tot} z(c(:, t)) \]

where \( z \) is a vector which changes with time:

\[z(c(:, t)) = \begin{bmatrix}
\frac{c_1 d_1}{3} & \frac{c_2 d_1}{6} & \frac{c_3 d_1}{6} & \cdots & \frac{c_N d_1}{6} \\
\frac{c_1 d_2}{3} & \frac{c_2 d_2}{6} & \frac{c_3 d_2}{6} & \cdots & \frac{c_N d_2}{6} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{c_1 d_M}{3} & \frac{c_2 d_M}{6} & \cdots & \frac{c_N d_M}{6}
\end{bmatrix}^T
\]

Finally we can proceed via backward Euler to solve (20) across time. We index time with \( j \):
The finite element discretization for the adjoint pde yields the same matrices than we had before:

\[
\begin{align*}
\mathbf{M} \frac{db}{dt} &= -D_b \mathbf{K} b - k_b^+ \mathbf{L}(c(\cdot, t)) b - k_b^- \mathbf{M} b + k_b^+ B_{tot} z(c(\cdot, t)) \\
\mathbf{M} \frac{b_j - b_{j-1}}{dt} &= -D_b \mathbf{K} b_j - k_b^+ \mathbf{L}(c_j) b_j - k_b^- \mathbf{M} b_j + k_b^+ B_{tot} z(c_j) \\
\mathbf{M} \frac{b_{j-1}}{dt} + k_b^+ B_{tot} z(c_j) &= \left( \frac{\mathbf{M}}{dt} + D_b \mathbf{K} + k_b^+ \mathbf{L}(c_j) + k_b^- \mathbf{M} \right) b_j \\
b_j &= \left( \frac{\mathbf{M}}{dt} + D_b \mathbf{K} + k_b^+ \mathbf{L}(c_j) + k_b^- \mathbf{M} \right)^{-1} \left( \frac{\mathbf{M} b_{j-1}}{dt} + k_b^+ B_{tot} z(c_j) \right)
\end{align*}
\]

Likewise, we do the same for the adjoint pde (11):

\[
D_b B_{xz} = -B_t + k_b^+ c B + k_b^- B + \sum_{j=1}^{N_x} \delta(x - x_j)(b - b^z)
\]

\[
\int_0^t D_b B_{xz} v dx = \int_0^t \left( -B_t + k_b^+ c B + k_b^- B + \sum_{j=1}^{N_x} \delta(x - x_j)(b - b^z) \right) v dx
\]

\[
\int_0^t D_b B_x v_n dx = \int_0^t B_t v dx - \int_0^t (k_b^+ c + k_b^-)^{-1} B v dx - \sum_{j=1}^{N_x} B_{tot}(b - b^z) v(x_j, t)
\]

now let \( B = \sum_{k=1}^{N_x} B_k(t) \phi_k(x), c = \sum_{k=1}^{N_x} c_k(t) \phi_k(x), v = \phi_n(x) \) for \( n = 1, 2, \ldots, N_x \) and we get:

\[
D_b \sum_{k=1}^{N_x} B_k(t) \int_0^t \phi_k \phi_n' dx = \sum_{k=1}^{N_x} \frac{\partial B_k}{\partial t}(t) \int_0^t \phi_k \phi_n dx - \sum_{j=1}^{N_x} \sum_{k=1}^{N_x} k_b^+ c_j B_k \int_0^t \phi_j \phi_k \phi_n dx
\]

\[
- \sum_{k=1}^{N_x} k_b^- B_k \int_0^t \phi_k \phi_n dx - \sum_{j=1}^{N_x} (b - b^z) \phi_n(x_j)
\]

The finite element discretization for the adjoint pde yields the same matrices than we had before:

\[
D_b \mathbf{K} B = \mathbf{M} \frac{dB}{dt} - k_b^+ \mathbf{L}(c(\cdot, t)) B - k_b^- \mathbf{M} B - q(t)
\]
\[
M \frac{dB}{dt} = -D_b KB - k_b^+ L(c(\cdot, t)) B - k_b^- MB - q \\
M \frac{B_j - B_{j-1}}{dt} = -D_b KB_{j-1} - k_b^+ L(c_{j-1}) B_{j-1} - k_b^- MB_{j-1} - q_{j-1} \\
B_{j-1} = \left( M - D_b K - k_b^+ L(c_{j-1}) - k_b^- M \right)^{-1} \left( \frac{M}{dt} B_j + q_{j-1} \right)
\]

3 Obtaining \( \nabla L(b, B, c) \)

Now with both \( b, B \) in hand, we turn our efforts towards computing the gradient. Let \( M_x = \{x_j\}_{j=1}^{N_x}, M_t = \{t_j\}_{j=1}^{N_t} \) be our sets of gridpoints for space and time respectively. Recall (5):

\[
(\partial \Phi(c), \tilde{c}) = (\partial_c L(c, b, B), \tilde{c}) = -\int_0^T \int_0^\ell k_b^+(B_{tot} - b) B \tilde{c} dx dt
\]

Now assume that our calcium parameter is piecewise linear in both space and time:

\[
c(x, t) = \sum_{k=1}^{N_x} \sum_{j=1}^{N_t} c_{k,j} H_k(x, M_x) H_j(t, M_t)
\]

Because Matlab’s `fminunc` expects a vector as the input parameter, we reindex our mesh so that we pass around our vector. If \( k = 1, 2, \ldots, N_x \) and \( j = 1, 2, \ldots, N_t \), then define

\[
i = N_t (k - 1) + j
\]

which in turn produces a new vector \( c_i = c_{k,j} \). Also take notice that for \( k, j \) in this range, each \( i \) produces a unique pair \((k, j)\). We will use these two indexing systems in our notation for clarity. Now finally we can write the gradient, after also assuming that our \( b, B \) are piecewise linear in both space and time:

\[
\frac{\partial \Phi(c)}{\partial c_i} = k_b^+ \int_0^T \int_0^\ell (B_{tot} - b) B H_k(x, M_x) H_j(t, M_t) dx dt
\]

\[
= k_b^+ \int_{t_{j-1}}^{t_{j+1}} \int_{x_{k-1}}^{x_{k+1}} (B_{tot} - b) B H_k(x, M_x) H_j(t, M_t) dx dt
\]

After substituting in

\[
b = \sum_{p=k-1}^{k+1} \sum_{q=j-1}^{j+1} b_{p,q} H_p(x, M_x) H_q(t, M_t)
\]

\[
B = \sum_{p=k-1}^{k+1} \sum_{q=j-1}^{j+1} B_{p,q} H_p(x, M_x) H_q(t, M_t)
\]
we can see that writing out (35) can become quite tedious. Instead, we leave it in its current form and do not explicitly write it out here, although one can see we have broken down the gradient into nothing more than a large sum of values coming from our known $b, B$ and $c$.

Finally, plug everything into a function, $\text{ObjFunVal}$, which returns both the value and gradient of our objective function:

$$[\text{val}, \text{grad}] = \text{ObjFunVal}$$

And call it in Matlab with something like this:

```matlab
fminunc('ObjFunVal', initguess, optimset('gradobj', 'on'))
```