Adjoint Method Approach For Calcium Equations

Ray Hwong, Samuel Feng

September 11, 2006

1 System and Gradient Derivation

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

$$\begin{aligned} c_t = D_c c_{xx} - k_b^+ c(B_{tot} - b) + k_b^- b + \alpha\delta(x - x_u)\delta(t - t_u) \\ b_t = D_b b_{xx} + k_b^+ c(B_{tot} - b) - k_b^- b \\ c_x(0, t) = c_x(\ell, t) = 0 \\ b_x(0, t) = b_x(\ell, t) = 0 \end{aligned} \tag{1}$$

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, α 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 $\{b^\sharp(x_j, t)\}_{j=1}^N$, over all possible calcium functions $c(x, t)$:

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

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

$$\begin{aligned} L(c, b, B) = \frac{1}{2} \sum_{j=1}^N \int_0^T \left(b^\sharp(x_j, t) - b(x_j, t)\right)^2 dt \\ + \int_0^T \int_0^\ell \left(b_t - D_b b_{xx} - k_b^+ c(B_{tot} - b) + k_b^- b\right) B dx dt \end{aligned} \tag{3}$$

It is important to take note that

$$\Phi(c) = \min_b \max_B L(c,b,B) \tag{4}$$

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

$$\langle \partial\Phi(c), \tilde{c}\rangle = \langle \partial_c L(c,b,B), \tilde{c}\rangle = -\int_0^T \int_0^\ell k_b^+ (B_{tot} - b) B\tilde{c} dxdt \tag{5}$$

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 $\left\langle \partial_b L(c,b,B), \tilde{b} \right\rangle = 0$ for all possible $\tilde{b}$:

$$\begin{aligned} 0 = \left\langle \partial_b L(c,b,B), \tilde{b} \right\rangle = \lim_{\varepsilon \to 0} & \frac{L(c, b - \varepsilon\tilde{b}, B) - L(c,b,B)}{\varepsilon} \\ &= \sum_{j=1}^N \int_0^T -(b^\sharp(x_j,t) - b(x_j,t))\tilde{b}(x_j,t)dt \\ &+ \int_0^T \int_0^\ell (\tilde{b}_t - D_b\tilde{b}_{xx} + k_b^+ c\tilde{b} + k_b^- \tilde{b}) B dxdt \end{aligned} \tag{6}$$

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:

$$\begin{aligned} \int_0^T \int_0^\ell \tilde{b}_t B dxdt = \int_0^\ell & \left[\tilde{b}B\right]_{t=0}^{t=T} dx - \int_0^T \int_0^\ell \tilde{b} B_t dxdt \\ &= \int_0^\ell \tilde{b}(x,T)B(x,T)dx - \int_0^T \int_0^\ell \tilde{b} B_t dxdt \end{aligned} \tag{7}$$

$$\begin{aligned} -D_b \int_0^T \int_0^\ell \tilde{b}_{xx} B dxdt &= -D_b \left(\int_0^T \left[\tilde{b}_x B\right]_{x=0}^{x=\ell} dt - \int_0^T \left[\tilde{b} B_x\right]_{x=0}^{x=\ell} dt + \int_0^T \int_0^\ell \tilde{b} B_{xx} dxdt \right) \\ &= D_b \left(\int_0^T \left[\tilde{b} B_x\right]_{x=0}^{x=\ell} dt + \int_0^T \int_0^\ell \tilde{b} B_{xx} dxdt \right) \end{aligned} \tag{8}$$

$$\sum_{j=1}^{N}\int_0^T -(b^\sharp(x_j,t)-b(x_j,t))\tilde{b}(x_j,t)dt = \int_0^T\int_0^\ell \sum_{j=1}^{N}\delta(x-x_j)(b-b^\sharp)\tilde{b}dxdt \tag{9}$$

Using these allows us to obtain our desired form:

$$\begin{aligned}0 = \left\langle \partial_b L(c,b,B),\tilde{b}\right\rangle \\ &= \int_0^T -D_b\tilde{b}(0,t)B_x(0,t)dt + \int_0^T D_b\tilde{b}(\ell,t)B_x(\ell,t)dt + \int_0^\ell \tilde{b}(x,T)B(x,T)dx \\ &\quad + \int_0^T\int_0^\ell (-B_t - D_bB_{xx} + k_b^+cB + k_b^-B + \sum_{j=1}^{N}\delta(x-x_j)(b-b^\sharp))\tilde{b}dxdt\end{aligned} \tag{10}$$

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

$$\begin{aligned}-B_t - D_bB_{xx} + (k_b^+c + k_b^-)B + \sum_{j=1}^{N}\delta(x-x_j)(b-b^\sharp) = 0 \\ B(x,T) = 0 \\ B_x(0,t) = B_x(\ell,t) = 0\end{aligned} \tag{11}$$

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 Φ 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 `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:

$$D_bb_{xx} = b_t - k_b^+cB_{tot} + (k_b^+ + k_b^-)b \tag{12}$$

$$\int_0^\ell D_bb_{xx}vdx = \int_0^\ell \left(b_t - k_b^+cB_{tot} + (k_b^+ - k_b^-)b\right)vdx \tag{13}$$

$$-\int_0^\ell D_bb_xv_xdx = \int_0^\ell \left(b_t - k_b^+cB_{tot} + (k_b^+ - k_b^-)b\right)vdx \tag{14}$$

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 ϕ_k's are the typical hat functions on our x grid) for $n = 1,2,\ldots,N_x$ we get the system:

$$
\begin{aligned}
-D_b \sum_{k=1}^{N_x} \int_0^{\ell} b_k(t)\phi_k'\phi_n' dx &= \int_0^{\ell} \sum_{k=1}^{N_x} \partial_t b_k \phi_k \phi_n dx \\
&+ \int_0^T \sum_{k=1}^{N_x}\sum_{j=1}^{N_x} k_b^+ c_k b_j \phi_k\phi_j\phi_n dx + \int_0^{\ell} \sum_{k=1}^{N_x} \left[k_b^- b_k - k_b^+ c_k\right]\phi_k\phi_n dx
\end{aligned} \tag{15}
$$

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:

$$
\mathbf{M} = \begin{bmatrix}
\frac{d_1}{3} & \frac{d_1}{6} & & & \\
\frac{d_1}{6} & \frac{d_1+d_2}{3} & \frac{d_2}{6} & & \\
& \ddots & \ddots & \ddots & \\
& & \frac{d_{N_x-2}}{6} & \frac{d_{N_x-2}+d_{N_x-1}}{3} & \frac{d_{N_x-1}}{6} \\
& & & \frac{d_{N_x-1}}{6} & \frac{d_{N_x-1}}{3}
\end{bmatrix} \tag{16}
$$

$$
\mathbf{K} = \begin{bmatrix}
\frac{1}{d_1} & -\frac{1}{d_1} & & & \\
-\frac{1}{d_1} & \frac{1}{d_1}+\frac{1}{d_2} & -\frac{1}{d_2} & & \\
& \ddots & \ddots & \ddots & \\
& & -\frac{1}{d_{N_x-2}} & \frac{1}{d_{N_x-2}}+\frac{1}{d_{N_x-1}} & -\frac{1}{d_{N_x-1}} \\
& & & -\frac{1}{d_{N_x-1}} & \frac{1}{d_{N_x-1}}
\end{bmatrix} \tag{17}
$$

(18)

For the next matrix, call $M = N_x - 1$ to conserve space:

$$
\mathbf{L}(c(:,t)) = \begin{bmatrix}
\frac{c_1 d_1}{4} + \frac{c_2 d_1}{12} & \frac{d_1(c_1+c_2)}{12} & & \\
\frac{d_1(c_1+c_2)}{12} & \frac{c_2(d_1+d_2)}{4} + \frac{c_1 d_1}{12} + \frac{c_3 d_2}{12} & \frac{d_2(c_2+c_3)}{12} & \\
& \ddots & \ddots & \ddots \\
& \frac{d_{N_x-2}(c_{N_x-2}+c_M)}{12} & \frac{c_M(d_{N_x-2}+d_M)}{4} + \frac{c_{N_x-2}d_{N_x-2}}{12} + \frac{c_{N_x}s_M}{12} & \frac{d_M(c_M+c_{N_x})}{12} \\
& & \frac{d_M(c_M+c_{N_x})}{12} & \frac{c_{N_x}d_M}{4} + \frac{c_M d_M}{12}
\end{bmatrix} \tag{19}
$$

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

$$
-D_b \mathbf{K} b = \mathbf{M}\frac{db}{dt} + k_b^+ \mathbf{L}(c(:,t))b + k_b^- \mathbf{M} b - k_b^+ B_{tot} z(c(:t)) \tag{20}
$$

where z is a vector which changes with time:

$$
z(c(:,t)) = \begin{bmatrix} \frac{c_1 d_1}{3} + \frac{c_2 d_1}{6} & \frac{c_1 d_1}{6} + \frac{c_2(d_1+d_2)}{3} + \frac{c_3 d_2}{6} & \cdots & \frac{c_{N_x} d_M}{2} \end{bmatrix}^T
$$

Finally we can proceed via backward Euler to solve (20) across time. We index time with j:

$$\mathbf{M}\frac{db}{dt} = -D_b\mathbf{K}b - k_b^+\mathbf{L}(c(:,t))b - k_b^-\mathbf{M}b + k_b^+B_{tot}z(c(:,t)) \tag{21}$$

$$\mathbf{M}\frac{b_j - b_{j-1}}{dt} = -D_b\mathbf{K}b_j - k_b^+\mathbf{L}(c_j)b_j - k_b^-\mathbf{M}b_j + k_b^+B_{tot}z(c_j) \tag{22}$$

$$\mathbf{M}\frac{b_{j-1}}{dt} + k_b^+B_{tot}z(c_j) = \left(\frac{\mathbf{M}}{dt} + D_b\mathbf{K} + k_b^+\mathbf{L}(c_j) + k_b^-\mathbf{M}\right)b_j \tag{23}$$

$$b_j = \left(\frac{\mathbf{M}}{dt} + D_b\mathbf{K} + k_b^+\mathbf{L}(c_j) + k_b^-\mathbf{M}\right)^{-1}\left(\frac{\mathbf{M}b_{j-1}}{dt} + k_b^+B_{tot}z(c_j)\right) \tag{24}$$

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

$$D_bB_{xx} = -B_t + k_b^+cB + k_b^-B + \sum_{j=1}^{N_x}\delta(x-x_j)(b-b^\sharp) \tag{25}$$

$$\int_0^\ell D_bB_{xx}v\,dx = \int_0^\ell\left(-B_t + k_b^+cB + k_b^-B + \sum_{j=1}^{N}\delta(x-x_j)(b-b^\sharp)\right)v\,dx \tag{26}$$

$$\int_0^\ell D_bB_xv_x\,dx = \int_0^\ell B_tv\,dx - \int_0^\ell\left(k_b^+c + k_b^-\right)Bv\,dx - \sum_{j=1}^{N_x}(b-b^\sharp)v(x_j,t) \tag{27}$$

$$\tag{28}$$

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:

$$\begin{aligned} D_b\sum_{k=1}^{N_x}B_k(t)\int_0^\ell \phi_k'\phi_n'\,dx = \sum_{k=1}^{N_x}\frac{\partial B_k}{\partial t}(t)\int_0^\ell \phi_k\phi_n\,dx - \sum_{j=1}^{N_x}\sum_{k=1}^{N_x}k_b^+c_jB_k\int_0^\ell \phi_j\phi_k\phi_n\,dx \\ - \sum_{k=1}^{N_x}k_b^-B_k\int_0^\ell \phi_k\phi_n\,dx - \sum_{j=1}^{N}(b-b^\sharp)\phi_n(x_j) \end{aligned} \tag{29}$$

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

$$D_b\mathbf{K}B = \mathbf{M}\frac{dB}{dt} - k_b^+\mathbf{L}(c(:,t))B - k_b^-\mathbf{M}B - q(t) \tag{30}$$

where q is a vector that changes with time, containing the difference between b and $b^\sharp$ at the gridpoints where $b^\sharp$ is defined and zeros otherwise. Again, we are now ready to proceed via Forward Euler (for stability), since now we solve backwards in time:

$$\mathbf{M}\frac{dB}{dt} = -D_b\mathbf{K}B - k_b^+\mathbf{L}(c(:,t))B - k_b^-\mathbf{M}B - q \quad (31)$$

$$\mathbf{M}\frac{B_j - B_{j-1}}{dt} = -D_b\mathbf{K}B_{j-1} - k_b^+\mathbf{L}(c_{j-1})B_{j-1} - k_b^-\mathbf{M}B_{j-1} - q_{j-1} \quad (32)$$

$$B_{j-1} = \left(\mathbf{M} - D_b\mathbf{K} - k_b^+\mathbf{L}(c_{j-1}) - k_b^-\mathbf{M}\right)^{-1}\left(\frac{\mathbf{M}}{dt}B_j + q_{j-1}\right) \quad (33)$$

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):

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

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

$$\frac{\partial\Phi(c)}{\partial c_i} = k_b^+\int_0^T\int_0^\ell (B_{tot} - b)BH_k(x, M_x)H_j(t, M_t)dxdt \quad (34)$$

$$= k_b^+\int_{t_{j-1}}^{t_{j+1}}\int_{x_{k-1}}^{x_{k+1}} (B_{tot} - b)BH_k(x, M_x)H_j(t, M_t)dxdt \quad (35)$$

$$(36)$$

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, `ObjFunVal`, which returns both the value and gradient of our objective function:

```
[val,grad] = ObjFunVal
```

And call it in Matlab with something like this:

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