Fast Methods Comp

Joshua T. Vogelstein, Baktash Babadi, Liam Paninski

February 19, 2008

Contents

Abstract

1 Introduction

The goal of this work is to infer the underlying spike train, having only the observed fluorescence signal available to us. To do so, we propose a parametric model of the experimental system. Then, we develop a number of algorithms, each iteratively estimating the parameters and then using those parameters to infer the hidden spike trains.

2 Methods

2.1 Model

First, for simplicity, we assume that we have extracted a time series of fluorescence observations, F_t, which is a function of the intracellular calcium concentration of the imaged soma at that time, $[\text{Ca}^{2+}]_t$. In particular, we assume that F_t simply a noisy version of $[\text{Ca}^{2+}]_t$, i.e.,

$$F_t = [\text{Ca}^{2+}]_t + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma^2) \tag{1}$$

where ε_t is an additive Gaussian noise term with zero mean and variance σ^2. Note that this time series is naturally discrete, with time step size Δ corresponding to the frame rate of the image acquisition system, with t indexing the time step, and T indicating the final time step. The $[\text{Ca}^{2+}]$ signal is assumed to jump immediate after each spike by A μM and then decay back to rest, $[\text{Ca}^{2+}]_0$, with time constant τ sec. Therefore, we have

$$[\text{Ca}^{2+}]_t = a[\text{Ca}^{2+}]_{t-1} + An_t + b, \tag{2}$$

where we have parameters $a = 1 - \frac{\Delta}{\tau}$ and $b = \frac{\Delta}{\tau}[\text{Ca}^{2+}]_0$, where $[\text{Ca}^{2+}]_0$ is the baseline $[\text{Ca}^{2+}]$, and n_t is the underlying spike train.

2.2 Inferring the spike times

We would like to find the most likely spike train that led to the observed fluorescence signal. This problem may be formally cast as a least-squares problem:

$$\vec{n}_{lsq} = \underset{\vec{n}}{\text{argmin}} \sum_{t=0}^{T} (F_t - [\text{Ca}^{2+}]_t)^2, \tag{3}$$

where we use the notation $\vec{n} = n_0, \ldots, n_T$, and $[\text{Ca}^{2+}]$ is implicitly a function of n. Figure 1 shows an example fluorescence times series (A), the true calcium signal (B), and the true spike train (C). As (3) is a quadratic optimization problem, it may be solved with a single Newton step (details for efficiently computing the necessary terms may be found in Appendix B). Figure 1(D) shows why the solution to (3) is problematic. First, in periods of quiescence (i.e., no spiking), the algorithm incorrectly infers the presence of many "spikelets", which do not exist in the true data. Second, the algorithm sometimes infers *negative* spikes, which also cannot exist. Third, the number of spikes in a frame is not necessarily an integer. Each of these issues may be treated using standard tools from the optimization literature.

To deal with the spikelets, we use a technique called "regularizing". Essentially, we impose a prior on the probability of their being a spike at any time, and then we penalize the solution if it infers too many spikes. Formally, we write:

$$\vec{n}_{reg} = \underset{\vec{n}}{\text{argmin}} \sum_{t=0}^{T} \big((F_t - [\text{Ca}^{2+}]_t)^2 + f_{reg}(n_t) \big), \tag{4}$$

where $f_{reg}(n_t)$ is some regularizing function of n_t, typically chosen such that the likelihood remains concave. For example, if $f_{reg}(n_t) = \mathcal{N}(n_t, \sigma_n^2)$, then we have a Gaussian regularizing term, and the optimal solution may be computed using the standard *Wiener deconvolution* approach (derived for this model in Appendix A). Alternately, if $f_{reg}(n_t) = \lambda_t n_t^2$, where $\lambda_t = 1/(\text{expected jump size at time } t)$, then we have the standard *ridge regression*. Figure 1(E) shows the output of the algorithm proposed previously by Yaksi and Friedrich (2006)**??**, which may be thought of as a custom approximation to either of these regularization approaches. Problematically, the solutions from such an approach possibly yield negative spikes, which are not possible in the true spike train. As such, it is of interest to find the best underlying *non-negative* spike train:

$$\vec{n}_{nng} = \underset{\vec{n}:n_t \geq 0}{\operatorname{argmin}} \sum_{t=0}^{T} \big((F_t - [\text{Ca}^{2+}]_t)^2 + \lambda_t n_t\big). \tag{5}$$

Note that we have chosen to let $f_{reg}(n_t) = \lambda n_t$. To efficiently solve this problem, we adopt the "barrier method". This approach iteratively solves a related problem that penalizes the algorithm for inferring negative values, but with each iteration, reduces the penalty, in such a way that is guaranteed to converge to (5)**??**. Formally, it may be written as

$$\vec{n}_\eta = \underset{\vec{n}}{\operatorname{argmin}} \sum_{t=0}^{T} \big((F_t - [\text{Ca}^{2+}]_t)^2 + \lambda_t n_t + \eta f(n_t)\big), \tag{6}$$

where $f(n_t)$ is a barrier function, which is any function that approaches infinity from the right as n_t approaches zero, e.g., $-\log(n_t)$. So, as $\eta \to 0$, $\vec{n}_\eta \to \vec{n}_{nng}$. The key to solving for $\vec{n}_\eta$ efficiently is that $\vec{n}_\eta$ may be written as a linear function of $[\text{Ca}^{2+}]_t$. In particular, after subtracting b, we may write:

$$\vec{n} = \boldsymbol{M}[\vec{\text{Ca}}^{2+}] \tag{7}$$

where $\boldsymbol{M}$ is a $T \times T$ bidiagonal deconvolution matrix:

$$\boldsymbol{M} = \begin{bmatrix} A & 0 & 0 & 0 & \dots & 0 \\ -a & A & 0 & 0 & \dots & 0 \\ 0 & -a & A & 0 & \dots & 0 \\ \vdots & & & & & \\ 0 & \dots & 0 & 0 & a & A \end{bmatrix}. \tag{8}$$

Because the likelihood $\mathcal{L}$ of $\vec{n}_\eta$ is log-concave, we can use the Newton approach, which requires computing the gradient $\vec{g}$ and Hessian $\boldsymbol{H}$, and stepping in the direction of $\boldsymbol{H}^{-1}\vec{g}$. Because $\boldsymbol{M}$ is bidiagonal, $\boldsymbol{H}$ is tridiagonal, so this computation may be performed in $O(T)$ time instead of the typical $O(T^3)$ time required for matrix inversions.

Figure 1(F) shows how this constraint further improves the inference accuracy, and Appendix B provides details for how to solve for $\vec{n}_\eta$ efficiently. Because of the barrier method's qualitative and quantitative improvement over the least-squares or regularized formulations, (3) and (4),

respectively, we do not consider them further. Although the barrier method provides us with a meaningful spike train inference, we can further constrain the inferred spike train to be integers, i.e.,

$$\vec{n}_{ppr} = \underset{\vec{n}:n_t\in 0,1,2,\ldots}{\operatorname{argmin}} \sum_{t=0}^{T} \big((F_t - [\mathrm{Ca}^{2+}]_t)^2 + \lambda_t n_t\big). \tag{9}$$

Note that this formulation obviates the need to impose a non-negative constraint, but the prior $\lambda_t n_t$ still performs a useful function. In particular, in noisy scenarios, it helps the algorithm not infer too many spikes. An efficient algorithm for solving (9) is called "projection pursuit regression" (PPR)**??**. Figure 1(G) shows the inferred spike train using this algorithm, and Appendix C provides details for how to solve for $\vec{n}_{ppr}$ efficiently. Although this approach imposes a desirable constraint (i.e., that spike trains include only integers), it also comes at a cost relative to the barrier method, as will be described in Section 3.

2.3 Inferring the parameters

All of the above approaches assuming a particular effect of spikes on $[\mathrm{Ca}^{2+}]$. More precisely, they all assume that after each spike, $[\mathrm{Ca}^{2+}]$ jumps by A μM and then decays back to $[\mathrm{Ca}^{2+}]_0$ μM with time constant τ sec. Let $[\widehat{\mathrm{Ca}}^{2+}]_t$ and $\widehat{n}_t$ be the estimated $[\mathrm{Ca}^{2+}]_t$ and n_t, respectively. We may then solve for:

$$\{\widehat{a}, \widehat{A}, \widehat{b}\} = \underset{a,A,b>0}{\operatorname{argmin}} \big(F_t - a[\widehat{\mathrm{Ca}}^{2+}]_{t-1} - A\widehat{n}_t - b\big)^2. \tag{10}$$

where, as before, we have $a = 1 - \frac{\Delta}{\tau}$ and $b = \frac{\Delta}{\tau}[\mathrm{Ca}^{2+}]_0$. This may be solved in matrix form by making the following substitutions:

$$[\vec{\mathrm{Ca}}^{2+}]_{est} = \begin{bmatrix} [\widehat{\mathrm{Ca}}^{2+}]_2 \\ \vdots \\ [\widehat{\mathrm{Ca}}^{2+}]_T \end{bmatrix}, \vec{n}_{est} = \begin{bmatrix} \widehat{n}_1 \\ \vdots \\ \widehat{n}_{T-1} \end{bmatrix}, \vec{F} = \begin{bmatrix} F_1 \\ \vdots \\ F_{T-1} \end{bmatrix}, \boldsymbol{G} = \begin{bmatrix} [\vec{\mathrm{Ca}}^{2+}]_{est} \\ \vec{n}_{est} \\ \vec{1} \end{bmatrix}, \quad \vec{x} = \begin{bmatrix} -a \\ -A \\ -b \end{bmatrix}, \tag{11}$$

where $\vec{1}$ is a column vector of ones of the appropriate length. Thus, we may write:

$$\widehat{x} = \underset{\vec{x}>\vec{0}}{\operatorname{argmin}} \left\| \boldsymbol{G}\vec{x} + \vec{F} \right\|_2^2 \tag{12}$$

$$= \underset{\vec{x}>\vec{0}}{\operatorname{argmin}} \frac{1}{T}\left(\frac{1}{2}\vec{x}'\boldsymbol{G}'\boldsymbol{G}\vec{x} + \boldsymbol{G}'\vec{F}\vec{x}\right) \tag{13}$$

which may be efficiently solved using, for instance, Matlab's `quadprog`. The variance of the error, σ^2, is simply the mean-squared-error of the residual**??**:

$$\sigma^2 = \frac{1}{T}\sum_{t=0}^{T}(F_t - [\widehat{\mathrm{Ca}}^{2+}]_t)^2 \tag{14}$$

3 Results

For certain scenarios, the desired temporal resolution of $[\mathrm{Ca}^{2+}]_t$ may be finer than Δ

A Wiener Deconvolution

We can write the above model as a convolution:

$$F(t) = y(t) * n(t) + \varepsilon(t) \tag{15}$$

where $y(t) = Ae^{-t/\tau}$ is the calcium convolution kernel. The goal is then to find some optimal deconvolution kernel, $g(t)$, so that we may estimate n_t as follows:

$$\widehat{n}(t) = g(t) * F(t). \tag{16}$$

The Wiener deconvolution filter is most easily described in the frequency domain:

$$G(f) = \frac{|Y^(f)|E[N(f)]^2}{|Y(f)|^2 E[N(f)]^2 + E[\epsilon(f)]^2} \tag{17}$$

where we use the notation $X(f)$ to indicate the Fourier transform of $x(t)$, and $E[X(f)]^2$ to indicate the power spectral density (PSD) of $x(t)$. By assuming that $n(t)$ is Poisson (or binomial), and $\varepsilon(t)$ is Gaussian, we can write their PSD's analytically:

$$E[N(f)]^2 = \tag{18}$$
$$E[\epsilon(f)]^2 = . \tag{19}$$

Similarly, because $y(t)$ is an exponential, we may write its Fourier transform as:

$$Y(f) = . \tag{20}$$

Thus, plugging in (18),(19), and (20) into (17), we have

$$G(f) = . \tag{21}$$

Finally, taking the inverse Fourier transform of $G(f)$, we are left with

$$g(t) =, \tag{22}$$

which we may plug back into (16) to find $\widehat{n}(t)$.

B Efficient Implementation of the Barrier Method

We are trying to solve (6) for a particular value of η. For each value of η, we can efficiently solve this problem using the Newton's method, as it is concave. First, we must compute the gradient (first derivative) and Hessian (second derivative) of the objective in (6) with respect to $[\text{Ca}^{2+}]$, which requires that we write n_t as a function of $[\text{Ca}^{2+}]_t$, which we can using (7). Then, we use these derivatives to choose the direction to ascend. The likelihood, gradient, and Hessian are, in matrix notation:

$$\mathcal{L} = \left\| \vec{F} - [\vec{\text{Ca}}^{2+}] \right\|_2^2 + \lambda \vec{1}' \vec{n} - \eta \log \vec{n} \tag{23}$$

$$= \left\| \vec{F} - [\vec{\text{Ca}}^{2+}] \right\|_2^2 + \lambda \vec{1}' \boldsymbol{M} [\vec{\text{Ca}}^{2+}] - \eta \log \boldsymbol{M} [\vec{\text{Ca}}^{2+}] \tag{24}$$

$$\vec{g} = -2(\vec{F} - [\vec{\text{Ca}}^{2+}]) + \lambda \vec{1}' \boldsymbol{M} - \eta \boldsymbol{M}' \log(\vec{n}^{-1}) \tag{25}$$

$$\boldsymbol{H} = 2 + 2\eta \boldsymbol{M}' \log(\vec{n}^{-2}) \boldsymbol{M} \tag{26}$$

Now the key is that the Hessian is a tridiagonal matrix, which follows from the fact that $\boldsymbol{M}$ is bidiagonal. Therefore, to compute the typical Newton step, update $[\vec{\text{Ca}}^{2+}]$ as follows:

$$[\vec{\text{Ca}}^{2+}] \leftarrow [\vec{\text{Ca}}^{2+}] + s\vec{d} \tag{27}$$

where $\vec{d} = \boldsymbol{H}^{-1} \vec{g}$, which may be performed in $O(T)$ time using standard efficient algorithms (e.g., Matlab's \), as opposed to the more general $O(T^3)$ time, due to the tridiagonal nature of $\boldsymbol{H}$. The step size s is typically taken to be unity, unless doing so violates any of the constraints. For this problem, we require that $n_t > 0$ for all t; therefore, we have:

$$0 < n_t \tag{28}$$

$$< \boldsymbol{M}(\vec{C} + s\vec{d}) \tag{29}$$

$$< \boldsymbol{M}\vec{C} + s\boldsymbol{M}\vec{d} \tag{30}$$

$$\Rightarrow s > -\boldsymbol{M}\vec{C}(\boldsymbol{M}\vec{d})^{-1}. \tag{31}$$

So s must be larger than every component of the vector on the right-hand-side of the last inequality. Furthermore, it is possible to "over-step", or rather, take a step so large as to *increase* the likelihood that we are trying to minimize (we climbed over the hill and started going back down again). As such, prior to stepping, we check that the likelihood would indeed decrease, if not, we reduce s until it does. Pseudocode for an efficient implementation of this approach is provided in Table 2. Note that to solve (3), $\vec{g}$ is simply $2([\vec{\text{Ca}}^{2+}] - \vec{F})$ and $\boldsymbol{H}$ is 2, and because the likelihood is quadratic, stepping once to $[\vec{\text{Ca}}^{2+}] + \boldsymbol{H}^{-1}\vec{g}$ yields $\vec{n}_{lsq}$.

C Projection Pursuit Method (PPR)

Table 1: Pseudocode for barrier method

For each η, let $[\vec{\text{Ca}}^{2+}]_i$ be the current estimate of $[\vec{\text{Ca}}^{2+}]$ (which may be initialized at $\vec{1}$ for instance). While $[\vec{\text{Ca}}^{2+}]_i < [\vec{\text{Ca}}^{2+}]_{i-1} + \xi$ (for some small ξ), iterate the following steps. When complete, reduce η and repeat.

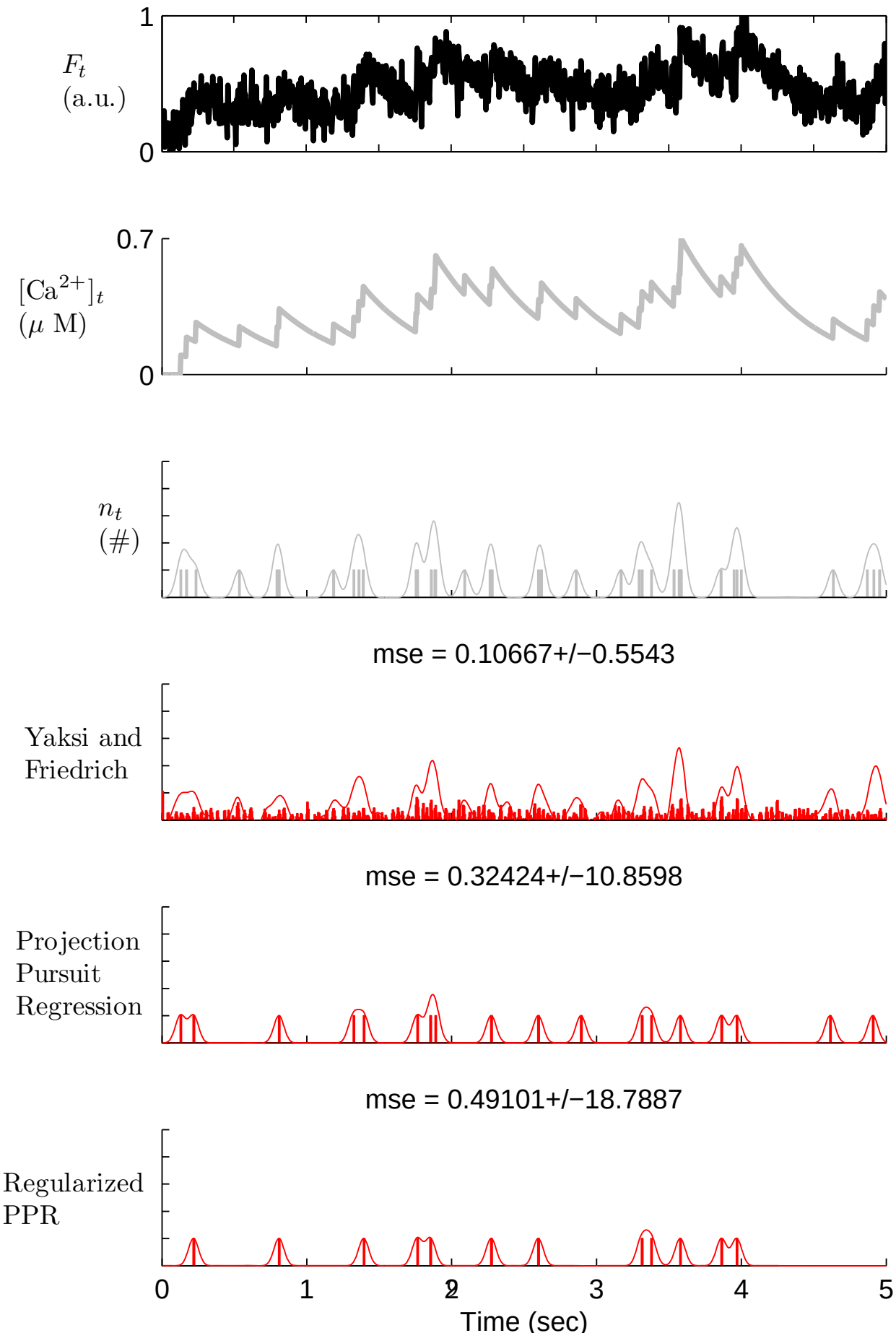
- Compute $\vec{g}$ using (25)
- Compute $\boldsymbol{H}$ using (26)
- Compute $\vec{d} = \boldsymbol{H}^{-1}\vec{g}$ efficiently
- Let $s = \min(1, 0.99 \min(-\boldsymbol{M}\vec{C}(\boldsymbol{M}\vec{d})^{-1})$
- Let $[\vec{\text{Ca}}^{2+}]_{temp} = [\vec{\text{Ca}}^{2+}]_i + s\vec{d}$
- If $\mathcal{L}([\vec{\text{Ca}}^{2+}]_{temp}) > \mathcal{L}([\vec{\text{Ca}}^{2+}]_i)$, then reduce s until the opposite it true. Else, $[\vec{\text{Ca}}^{2+}]_{i+1} = [\vec{\text{Ca}}^{2+}]_{temp}$.

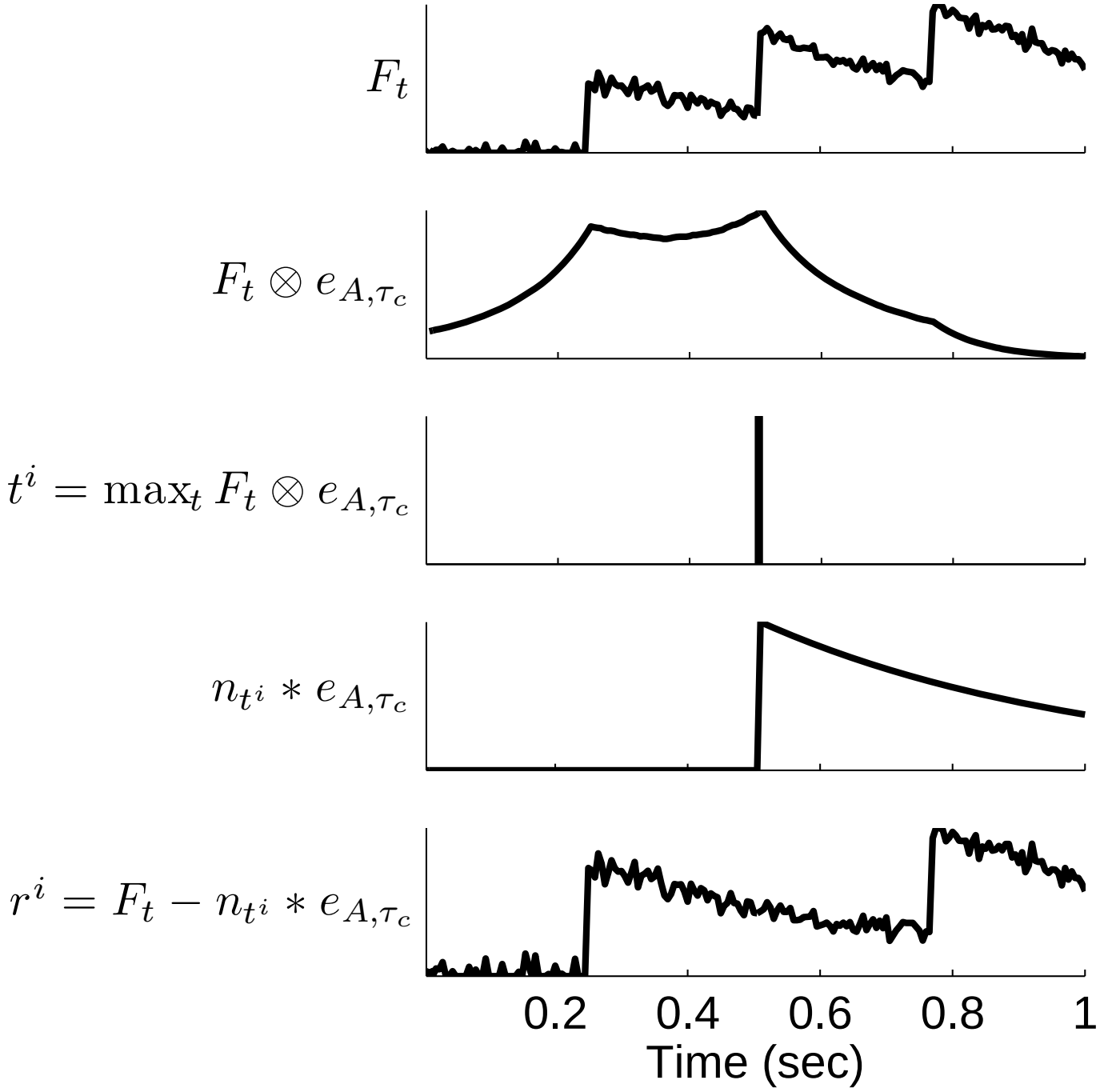
Table 2: Pseudocode for PPR

This algorithm iterates several steps, as schematized by Figure **??**. The algorithm is initialized with the fluorescence signal, which is the residual before iterating the following steps, $\vec{r}_0$:

- Compute the cross-correlation between the residual and the calcium kernel, $\vec{r}_0 \otimes Ae^{-t/\tau}$.
- Let t_{i+1} be the maximum of that cross-correlation, $t_{i+1} = \max_t \vec{r}_0 \otimes Ae^{-t/\tau}$.
- Convolve the calcium kernel with t_i to generate the expected effect of a spike having occurred at that time step, $t_{i+1} * e^{-t/\tau}$.
- Subtract the result of (**??**) from the residual of the previous step to get the new residual, $r_{i+1} = r_i - t_{i+1} * e^{-t/\tau}$
- Compute the regularized sum of residual error, where $n_{i+1,t}$ is a vector of zeros with ones for each $t = t_i$, $\epsilon_{i+1} = \sum_t r_{i+1,t} + \lambda_t n_{i+1,t}$
- If $r_{i+1} < r_i$, repeat. Otherwise, let $\vec{n}_{ppr} = \vec{n}_i$.

D Figures


F_t
(a.u.)
1
0
$[Ca^{2+}]_t$
(μ M)
0.7
0
n_t
(#)
mse = 0.10667+/−0.5543
Yaksi and
Friedrich
mse = 0.32424+/−10.8598
Projection
Pursuit
Regression
mse = 0.49101+/−18.7887
Regularized
PPR
0
1
2
3
4
5
Time (sec)


F_t
$F_t \otimes e_{A,\tau_c}$
$t^i = \max_t F_t \otimes e_{A,\tau_c}$
$n_{t^i} * e_{A,\tau_c}$
$r^i = F_t - n_{t^i} * e_{A,\tau_c}$
0.2
0.4
0.6
0.8
1
Time (sec)