A new look at state-space models for neural data

Liam Paninski

Department of Statistics and Center for Theoretical Neuroscience
Columbia University
http://www.stat.columbia.edu/∼liam
liam@stat.columbia.edu

March 22, 2010

Support: CRCNS, Sloan Fellowship, NSF CAREER, McKnight Scholar award.

State-space models

Unobserved state q_t with Markov dynamics $p(q_{t+1}|q_t)$

Observed y_t: $p(y_t|q_t)$

Goal: infer $p(q_t|Y_{0:T})$

Exact solutions: finite state-space HMM, Kalman filter (KF): forward-backward algorithm (recursive; $O(T)$ time)

Approximate solutions: extended KF, particle filter, etc.... basic idea: recursively update an approximation to "forward" distribution $p(q_t|Y_{0:t})$

Computing the MAP path

We often want to compute the MAP estimate

$$\hat{Q} = \arg\max_{Q} p(Q|Y).$$

In standard Kalman setting, forward-backward gives MAP (because $E(Q|Y)$ and $\hat{Q}$ coincide in Gaussian case).

More generally, extended Kalman-based methods give approximate MAP, but are non-robust: forward distribution $p(q_t|Y_{0:t})$ may be highly non-Gaussian even if full joint distribution $p(Q|Y)$ is nice and log-concave.

Write out the posterior:

$$\begin{aligned} \log p(Q|Y) &= \log p(Q) + \log p(Y|Q) \\ &= \sum_t \log p(q_{t+1}|q_t) + \sum_t \log p(y_t|q_t) \end{aligned}$$

Two basic observations:

- If $\log p(q_{t+1}|q_t)$ and $\log p(y_t|q_t)$ are concave, then so is $\log p(Q|Y)$.
- Hessian H of $\log p(Q|Y)$ is block-tridiagonal: $p(y_t|q_t)$ contributes a block-diag term, and $\log p(q_{t+1}|q_t)$ contributes a block-tridiag term.

Now recall Newton's method: iteratively solve $HQ_{dir} = \nabla$. Solving tridiagonal systems requires $O(T)$ time.

— computing MAP by Newton's method requires $O(T)$ time, even in highly non-Gaussian cases.

Newton here acts as an iteratively reweighted Kalman smoother (Fahrmeir and Kaufmann, 1991; Davis and Rodriguez-Yam, 2005; Jungbacker and Koopman, 2007); all suff. stats may be obtained in $O(T)$ time.

Parameter estimation

Standard method: Expectation-Maximization (EM). Iterate between computing $E(Q|Y)$ (or $\hat{Q}$) and maximizing w.r.t. parameters θ. Can be seen as coordinate ascent (slow) on first two terms of Laplace approximation:

$$\begin{aligned}
\log p(Y|\theta) &= \log \int p(Q|\theta)p(Y|\theta,Q)dQ \\
&\approx \log p(\hat{Q}_\theta|\theta) + \log p(Y|\hat{Q}_\theta,\theta) - \frac{1}{2}\log|H_{\hat{Q}_\theta}| \\
\hat{Q}_\theta &= \arg\max_Q \{\log p(Q|\theta) + \log p(Y|Q,\theta)\}
\end{aligned}$$

Faster: simultaneous joint optimization.

$$\begin{aligned}
\hat{\theta} &= \arg\max_\theta \left\{\log p(\hat{Q}_\theta|\theta) + \log p(Y|\hat{Q}_\theta,\theta)\right\} \\
&= \arg\max_\theta \max_Q \left\{\log p(\hat{Q}|\theta) + \log p(Y|\hat{Q},\theta)\right\}.
\end{aligned}$$

Write the joint Hessian in (Q,θ) as $\begin{pmatrix} H_{\theta\theta} & H_{\theta Q}^T \\ H_{\theta Q} & H_{QQ} \end{pmatrix}$, with H_{QQ} block-tridiag.

Now use the Schur complement to efficiently compute the Newton step (Koyama and Paninski, 2009; Paninski et al., 2010).

Computing $\nabla_\theta \log|H_{\hat{Q}_\theta}|$ also turns out to be easy ($O(T)$ time) here.

Constrained optimization

In many cases we need to impose constraints (e.g., nonnegativity) on q_t. Easy to incorporate here, via interior-point (barrier) methods:

$$\begin{aligned}
\arg\max_{Q \in C} \log p(Q|Y) &= \lim_{\epsilon \searrow 0} \arg\max_{Q} \left\{ \log p(Q|Y) + \epsilon \sum_t f(q_t) \right\} \\
&= \lim_{\epsilon \searrow 0} \arg\max_{Q} \left\{ \sum_t \log p(q_{t+1}|q_t) + \log p(y_t|q_t) + \epsilon f(q_t) \right\};
\end{aligned}$$

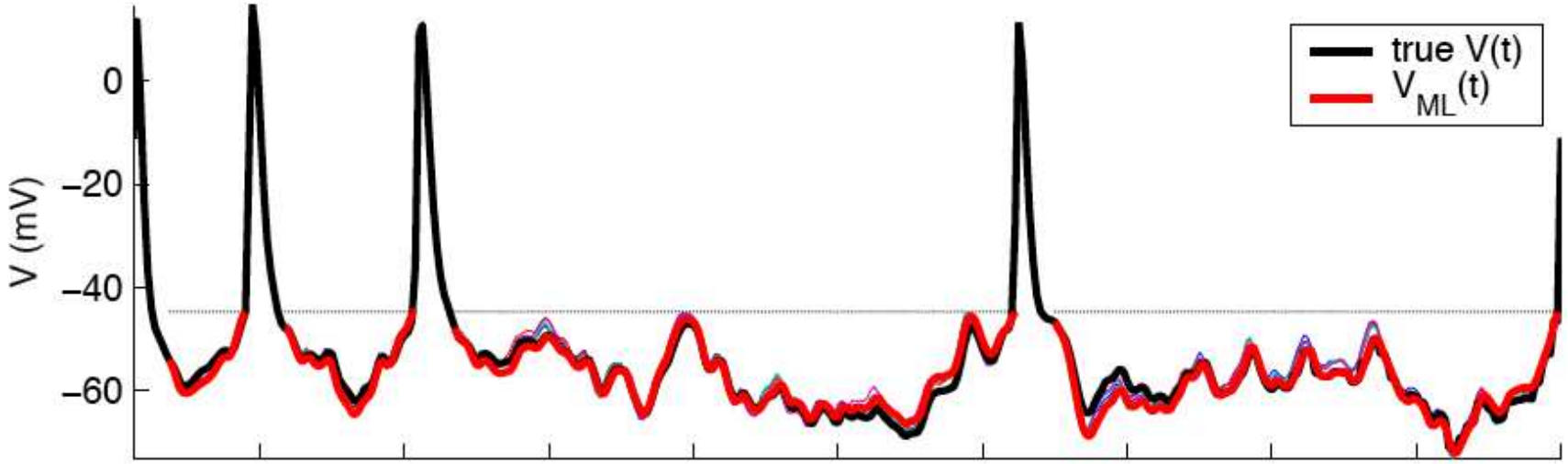
$f(.)$ is concave and approaching $-\infty$ near boundary of constraint set C. The Hessian remains block-tridiagonal and negative semidefinite for all $\epsilon > 0$, so optimization still requires just $O(T)$ time.

Example: computing the MAP subthreshold voltage given superthreshold spikes

Leaky, noisy integrate-and-fire model:

$$V(t+dt) = V(t) - dtV(t)/\tau + \sigma\sqrt{dt}\epsilon_t, \;\; \epsilon_t \sim \mathcal{N}(0,1)$$

Observations: $y_t = 0$ (no spike) if $V_t < V_{th}$; $y_t = 1$ if $V_t = V_{th}$

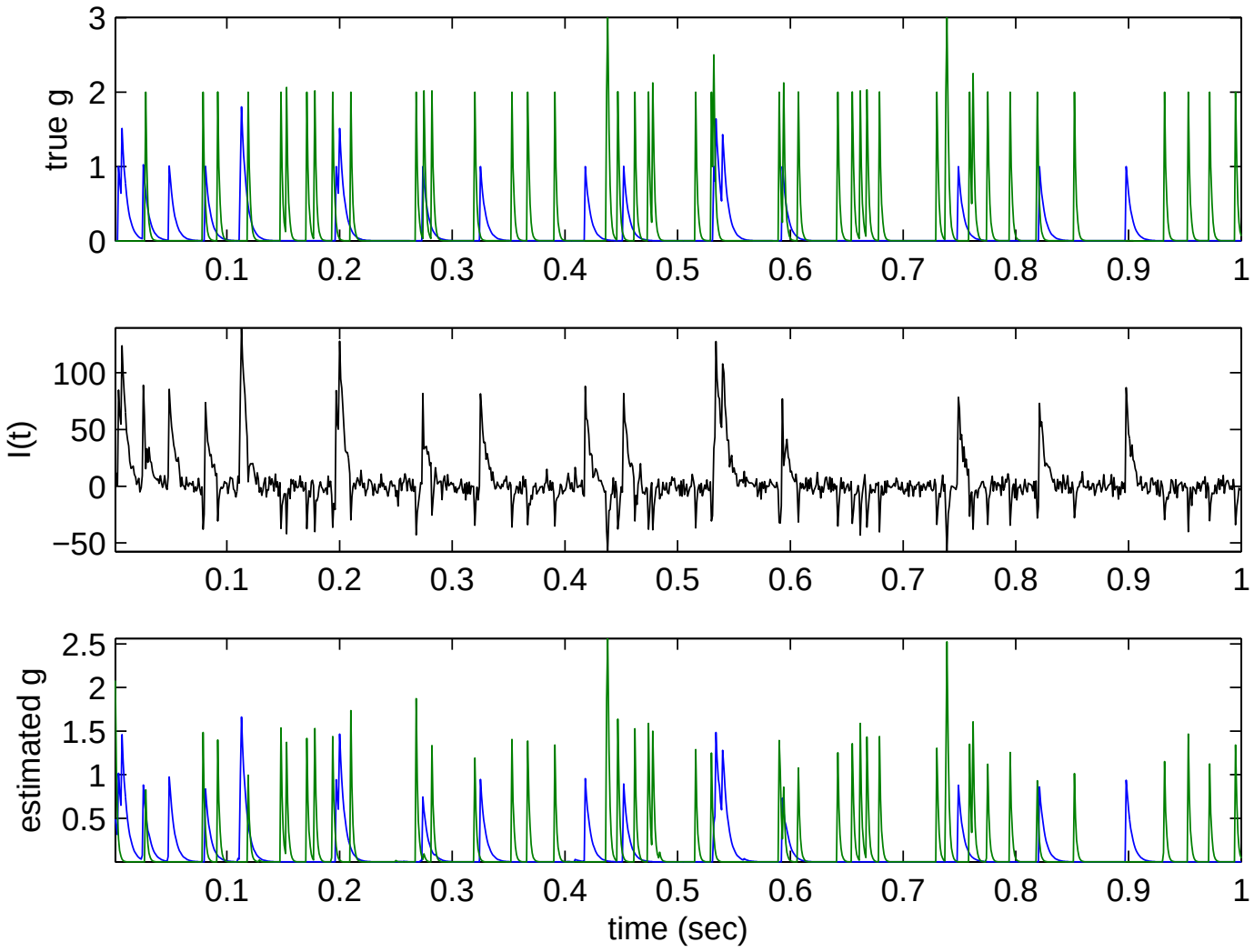


(Paninski, 2006)

Example: inferring presynaptic input

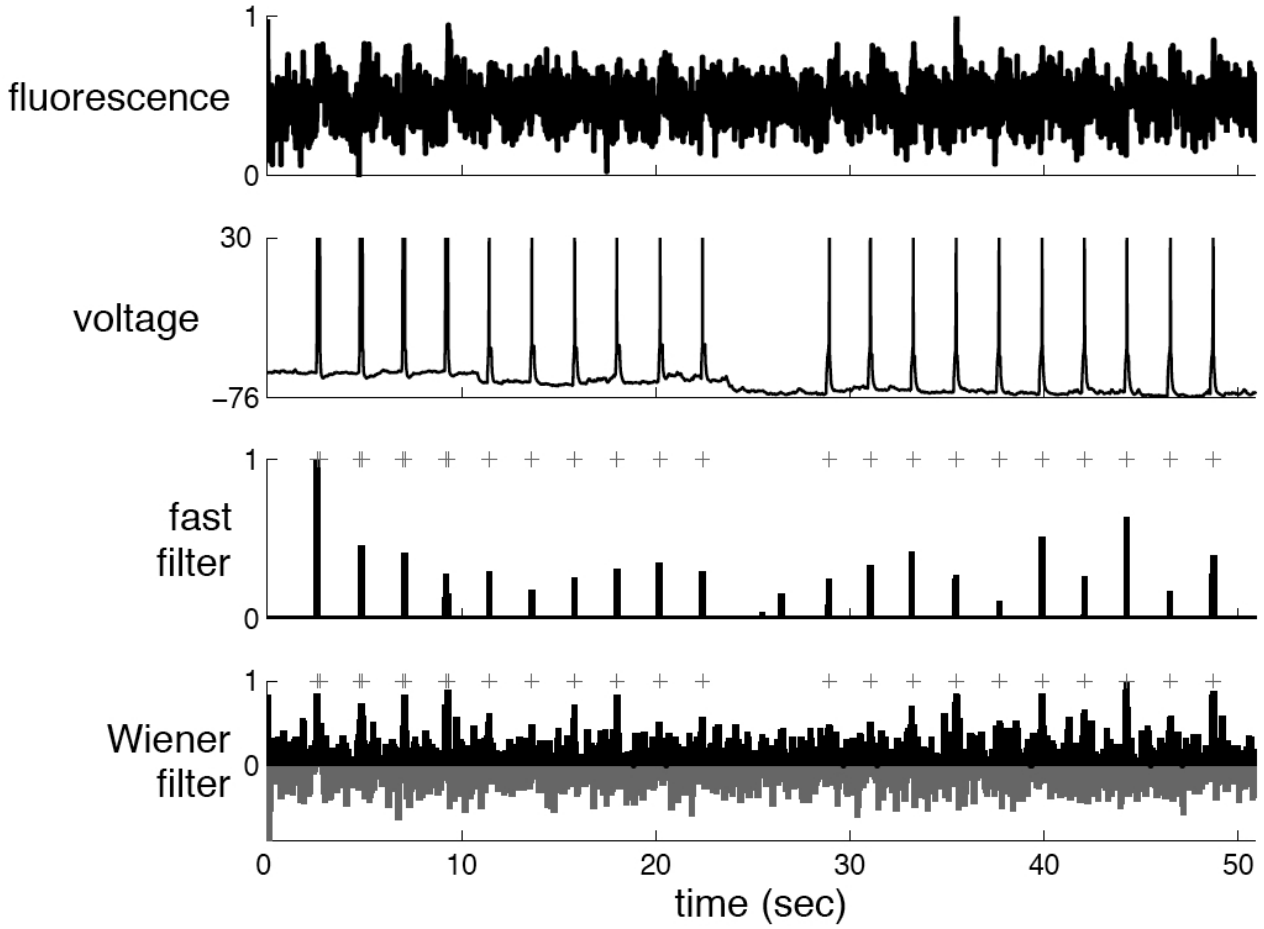
$$I_t = \sum_j g_j(t)(V_j - V_t)$$

$$g_j(t+dt) = g_j(t) - dt g_j(t)/\tau_j + N_j(t),\ N_j(t) > 0$$



Example: inferring spike times from slow, noisy calcium data

$$C(t+dt) = C(t) - dtC(t)/\tau + N_t;\ N_t > 0;\ y_t = C_t + \epsilon_t$$



— nonnegative deconvolution is a recurring problem in signal processing; many other possible applications (Vogelstein et al., 2008).

Further generalizations: GLM spike train decoding

We've emphasized tridiagonal structure so far, but similar results hold for any problem with a banded Hessian.

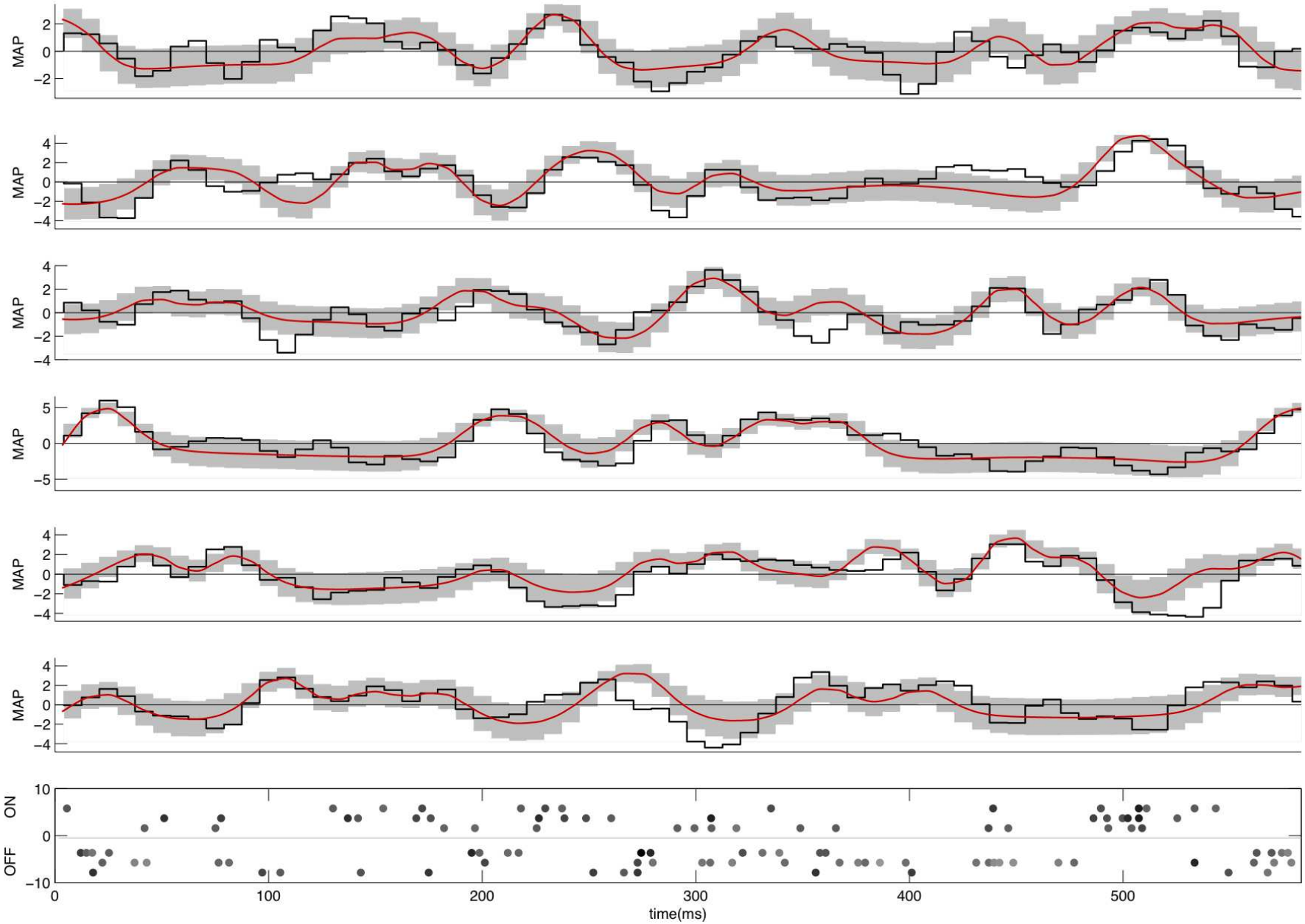
For example, look at point-process GLM:

$$\lambda_i(t) = f\left[b + \vec{k}_i \cdot \vec{x}(t) + \sum_{i',j} h_{i',j} n_{i'}(t-j)\right]$$

If the spatiotemporal filter $\vec{k}_i$ has a finite impulse response, then Hessian (w.r.t. $\vec{x}(t)$) is banded and optimal decoding of stimulus $\vec{x}(t)$ requires $O(T)$ time.

Similar speedups for MCMC methods (Ahmadian et al., 2010).

How important is timing?



(Ahmadian et al., 2010)

Optimal control of spike timing

Optimal experimental design and neural prosthetics applications require us to perturb the network at will. How can we make a neuron fire exactly when we want it to?

Assume bounded inputs; otherwise problem is trivial.

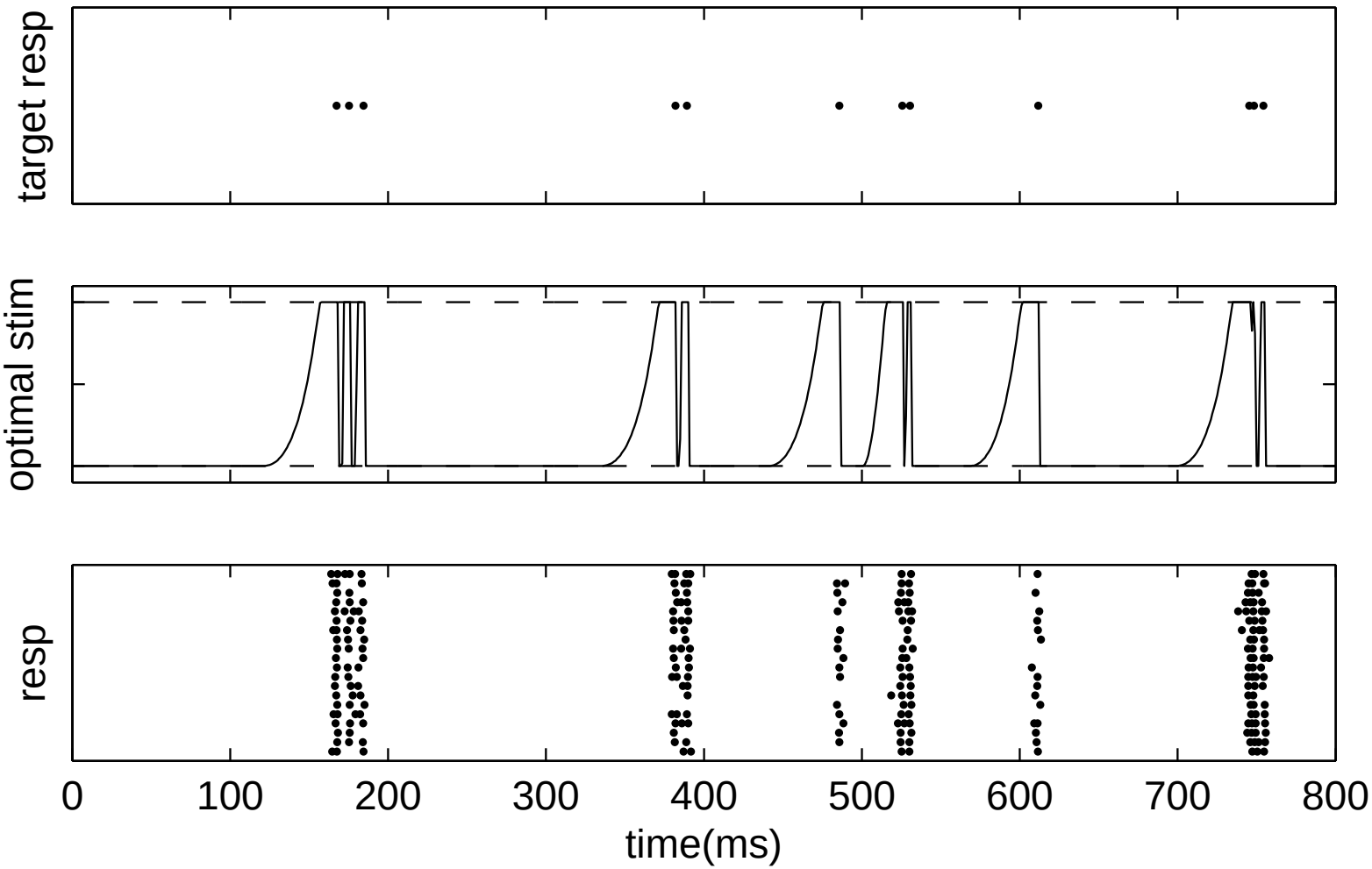
Start with a simple model:

$$\lambda_t \quad = \quad f(\vec{k} * I_t + h_t).$$

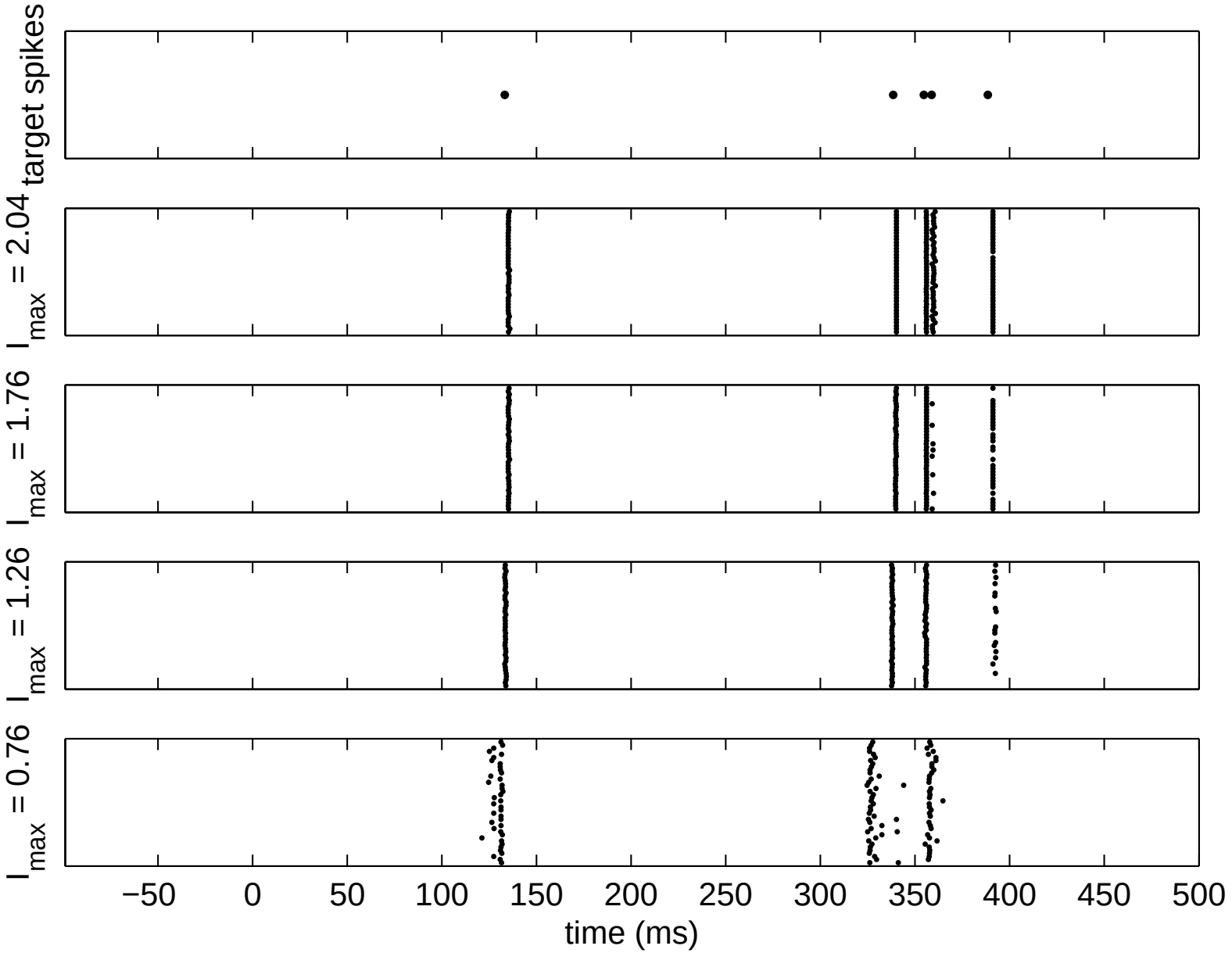
Now we can just optimize the likelihood of the desired spike train, as a function of the input I_t, with I_t bounded.

Concave objective function over convex set of possible inputs I_t + Hessian is banded $\implies$ $O(T)$ optimization.

Optimal electrical control of spike timing



Example: intracellular control of spike timing

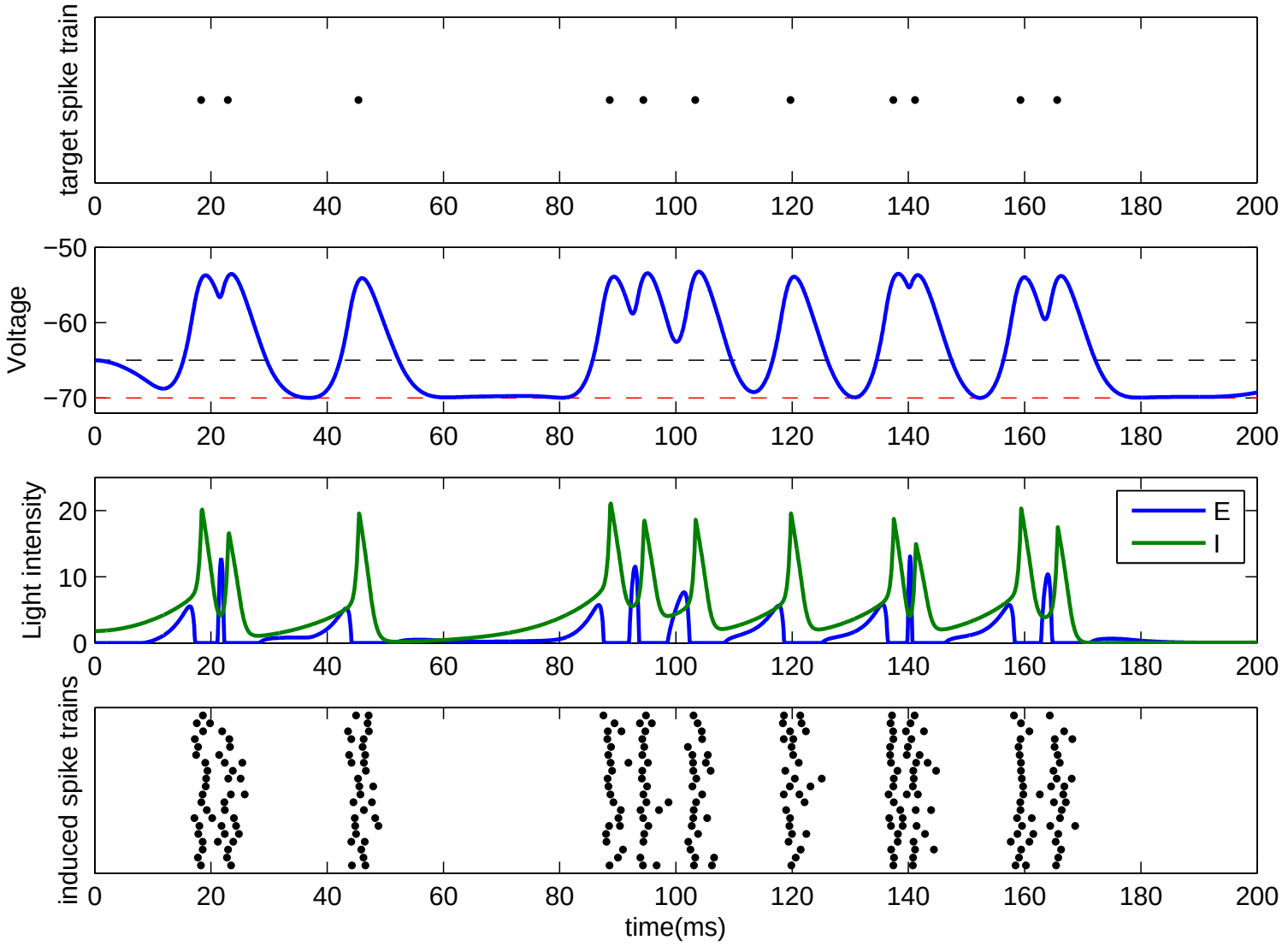


(Ahmadian et al 2010)

Optical conductance-based control of spiking

$$V_{t+dt} = V_t + dt\left(-gV_t + g_t^i(V^i - V_t) + g_t^e(V^e - V_t)\right) + \sqrt{dt}\sigma\epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0,1)$$

$$g_{t+dt}^i = g_t^i + dt\left(-\frac{g_t^i}{\tau_i} + a_{ii}L_t^i + a_{ie}L_t^e\right);\ g_{t+dt}^e = g_t^e + dt\left(-\frac{g_t^e}{\tau_i} + a_{ee}L_t^e + a_{ei}L_t^i\right)$$



One last extension: two-d smoothing

Estimation of two-d firing rate surfaces comes up in a number of examples:

- place fields / grid cells
- post-fitting in spike-triggered covariance analysis
- tracking of non-stationary (time-varying) tuning curves
- "inhomogeneous Markov interval" models for spike-history dependence

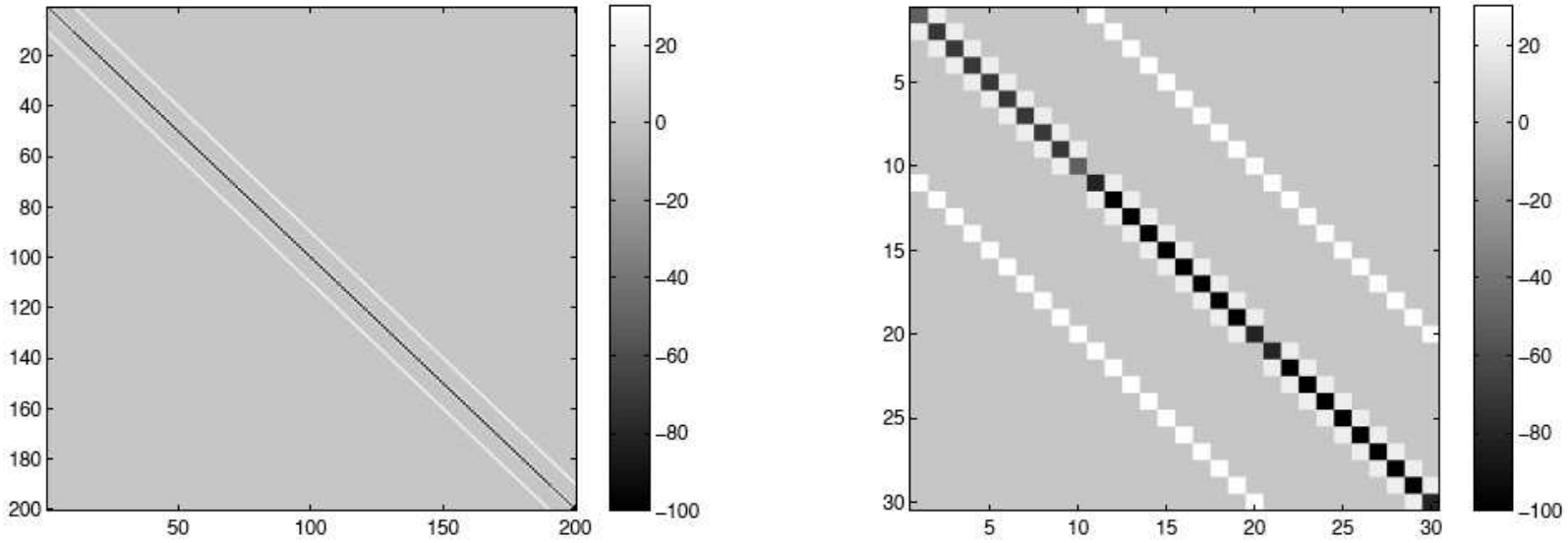
How to generalize fast 1-d state-space methods to 2-d case? Idea: use Gaussian process priors which are carefully selected to give banded Hessians.

Model: hidden variable Q is a random surface with a Gaussian prior:
$Q \sim \mathcal{N}(\mu, C)$;

Spikes are generated by a point process whose rate is a function of Q:
$\lambda(\vec{x}) = f[Q(\vec{x})]$ (easy to incorporate additional effects here, e.g. spike history)

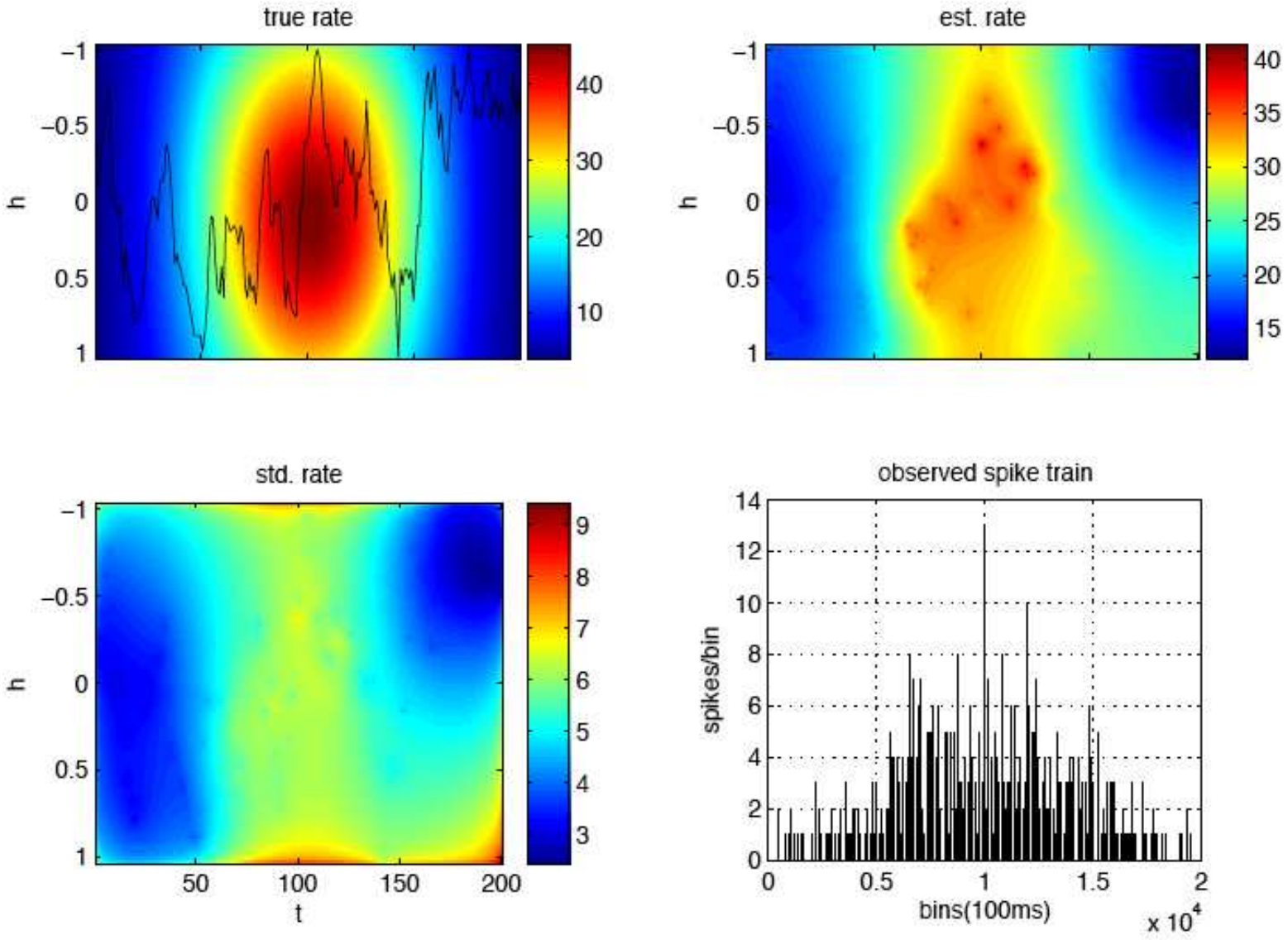
Now the Hessian of the log-posterior of Q is $C^{-1} + D$, where D is diagonal (Cunningham et al., 2007). For Newton, we need to solve $(C^{-1} + D)Q_{dir} = \nabla$.

Example: nearest-neighbor smoothing prior



For prior covariance C such that C^{-1} contains only neighbor potentials, we can solve $(C^{-1} + D)Q_{dir} = \nabla$ in $O(\dim(Q)^{1.5})$ time using direct methods ("approximate minimum degree" algorithm — built-in to Matlab sparse $A\backslash b$ code) and potentially in $O(\dim(Q))$ time using multigrid (iterative) methods (Rahnama Rad and Paninski, 2009).

Estimating a time-varying tuning curve given a limited sample path



Conclusions

- GLM and state-space approaches provide flexible, powerful methods for answering key questions in neuroscience
- Close relationships between forward-backward methods familiar from state-space theory and banded matrices familiar from spline theory
- Log-concavity, banded matrix methods make computations very tractable

References

Ahmadian, Y., Pillow, J., and Paninski, L. (2010). Efficient Markov Chain Monte Carlo methods for decoding population spike trains. *Under review, Neural Computation*.

Cunningham, J., Yu, B., Shenoy, K., and Sahani, M. (2007). Inferring neural firing rates from spike trains using Gaussian processes. *NIPS*.

Davis, R. and Rodriguez-Yam, G. (2005). Estimation for state-space models: an approximate likelihood approach. *Statistica Sinica*, 15:381–406.

Fahrmeir, L. and Kaufmann, H. (1991). On Kalman filtering, posterior mode estimation and fisher scoring in dynamic exponential family regression. *Metrika*, 38:37–60.

Jungbacker, B. and Koopman, S. (2007). Monte Carlo estimation for nonlinear non-Gaussian state space models. *Biometrika*, 94:827–839.

Koyama, S. and Paninski, L. (2009). Efficient computation of the maximum a posteriori path and parameter estimation in integrate-and-fire and more general state-space models. *Journal of Computational Neuroscience*, In press.

Paninski, L. (2006). The most likely voltage path and large deviations approximations for integrate-and-fire neurons. *Journal of Computational Neuroscience*, 21:71–87.

Paninski, L. (2010). Inferring synaptic inputs given a noisy voltage trace via sequential Monte Carlo methods. *Journal of Computational Neuroscience*, Under review.

Paninski, L., Ahmadian, Y., Ferreira, D., Koyama, S., Rahnama, K., Vidne, M., Vogelstein, J., and Wu, W. (2010). A new look at state-space models for neural data. *Journal of Computational Neuroscience*, In press.

Rahnama Rad, K. and Paninski, L. (2009). Efficient estimation of two-dimensional firing rate surfaces via Gaussian process methods. *Under review*.

Vogelstein, J., Babadi, B., Watson, B., Yuste, R., and Paninski, L. (2008). Fast nonnegative deconvolution via tridiagonal interior-point methods, applied to calcium fluorescence data. *Statistical analysis of neural data (SAND) conference*.