APPENDICES

A MODEL EXTENSIONS

A.1 Multidimensional Time Series

Although this paper only considers examples in which each $x_t^{(n)} \in \mathbb{R}$ and each $y_t^{(n)}$ is one-dimensional, our model and inference algorithm can be extended to cases in which observed time series and/or latent states have multiple dimensions. For example, as demonstrated in Section 5.2 of (Heng et al., 2017), cSMC scales well with a 64-dimensional vector time series model, suggesting that our proposed clustering approach with particle filtering is also applicable to multivariate series.

A.2 Finite Mixture of Time Series

It is simple to convert our model into one in which the true number of clusters K is known a priori. Instead of using a Dirichlet process, we can simply use a Dirichlet($\boldsymbol{\alpha}$) distribution in which $\boldsymbol{\alpha}$ is a K-dimensional vector with each $\alpha^{(k)} > 0$ for $k = 1, \ldots, K$. Then, we can modify Equation 2 as:

$$\begin{aligned} \boldsymbol{q} \mid \boldsymbol{\alpha} &\sim \text{Dirichlet}(\boldsymbol{\alpha}), \\ z^{(1)}, \ldots, z^{(N)} \mid \boldsymbol{q} &\sim \text{Multinomial}(N, \boldsymbol{q}), \\ \boldsymbol{\theta}^{(k)} \mid G &\sim G, && 1 \leq k \leq K, \end{aligned}$$

where $\boldsymbol{q}$ is an intermediary variable that is easy to integrate over.

The resultant inference algorithm is simpler. The only necessary modification to Algorithm 1 is that, when sampling cluster assignments, there is no longer any need for an auxiliary integer parameter $m \geq 1$ to represent the infinite mixture. Thus, Equation 6 becomes

$$p(z^{(n)} = k \mid Z^{(-n)}, \boldsymbol{\alpha}) = \frac{N^{(k)}}{N - 1 + \alpha^{(k)}}, \qquad k = 1, \ldots, K,$$

where $N^{(k)}$ is the number of cluster assignments equal to k in $Z^{(-n)}$. The process of sampling cluster parameters remains exactly the same as in the infinite mixture case.

B CONTROLLED SEQUENTIAL MONTE CARLO

A key step in sampling both the cluster assignments and the cluster parameters of Algorithm 1 is computing the parameter likelihood $p(\boldsymbol{y} \mid \boldsymbol{\theta})$ for an observation vector $\boldsymbol{y} = y_1, \ldots, y_T$ and a given set of parameters $\boldsymbol{\theta}$.

Recall the state-space model formulation:

$$\begin{aligned} x_1 \mid \boldsymbol{\theta} &\sim h(x_1; \boldsymbol{\theta}), \\ x_t \mid x_{t-1}, \boldsymbol{\theta} &\sim f(x_{t-1}, x_t; \boldsymbol{\theta}), && 1 < t \leq T, \\ y_t \mid x_t, \boldsymbol{\theta} &\sim g(x_t, y_t; \boldsymbol{\theta}), && 1 \leq t \leq T. \end{aligned}$$

B.1 Bootstrap Particle Filter

The bootstrap particle filter (BPF) of Doucet et al. (2001) is based on a sequential importance sampling procedure that iteratively approximates each filtering distribution $p(x_t \mid y_1, \ldots, y_t, \boldsymbol{\theta})$ with a set of S particles $\{x_t^1, \ldots, x_t^S\}$ so that

$$\hat{p}(\boldsymbol{y} \mid \boldsymbol{\theta}) = \prod_{t=1}^{T} \left(\frac{1}{S} \sum_{s=1}^{S} g(x_t^s, y_t; \boldsymbol{\theta}) \right)$$

is an unbiased estimate of the parameter likelihood $p(\boldsymbol{y} \mid \boldsymbol{\theta})$. Algorithm 2 provides a review of this algorithm.

Algorithm 2 `BootstrapParticleFilter`($\boldsymbol{y}$, $\boldsymbol{\theta}$, f, g, h)

1: **for** $s = 1, \ldots, S$ **do**
2: Sample $x_1^s \sim h(x_1; \boldsymbol{\theta})$ and weight $w_1^s = g(x_1^s, y_1; \boldsymbol{\theta})$.
3: **end for**
4: Normalize $\{w_1^s\}_{s=1}^S = \{w_1^s\}_{s=1}^S / \sum_{s=1}^S w_1^s$.
5: **for** $t = 2, \ldots, T$ **do**
6: **for** $s = 1, \ldots, S$ **do**
7: Resample ancestor index $a \sim \text{Categorical}(w_{t-1}^1, \ldots, w_{t-1}^S)$.
8: Sample $x_t^s \sim f(x_{t-1}^a, x_t; \boldsymbol{\theta})$ and weight $w_t^s = g(x_t^s, y_t; \boldsymbol{\theta})$.
9: **end for**
10: Normalize $\{w_t^s\}_{s=1}^S = \{w_t^s\}_{s=1}^S / \sum_{s=1}^S w_t^s$.
11: **end for**
12: **return** Particles $\{\{x_1^s\}_{s=1}^S, \ldots, \{x_T^s\}_{s=1}^S\}$

There are a variety of algorithms for the resampling step of Line 7. We use the systematic resampling method.

A common problem with the BPF is that although its estimate of $p(\boldsymbol{y} \mid \boldsymbol{\theta})$ is unbiased, this approximation may have high variance for certain observation vectors $\boldsymbol{y}$. The variance can be reduced at the price of increasing the number of particles, yet this often significantly increases computation time and is therefore unsatisfactory. To remedy our problem, we follow the work of Heng et al. (2017) in using controlled sequential Monte Carlo (cSMC) as an alternative to the standard bootstrap particle filter.

B.2 Twisted Sequential Monte Carlo

The basic idea of cSMC is to run several iterations of twisted sequential Monte Carlo, a process in which we redefine the model's state transition density f, initial prior h, and state-dependent likelihood g in a way that allows the BPF to produce lower-variance estimates without changing the parameter likelihood $p(\boldsymbol{y} \mid \boldsymbol{\theta})$. See also Guarniero et al. (2017) for a different iterative approach. Using a *policy* $\gamma = \{\gamma_1, \ldots, \gamma_T\}$ in which each γ_t is a positive and bounded function, we define,

$$
\begin{aligned}
h^\gamma(x_1; \boldsymbol{\theta}) &= \frac{h(x_1; \boldsymbol{\theta}) \cdot \gamma_1(x_1)}{H^\gamma(\boldsymbol{\theta})}, \\
f_t^\gamma(x_{t-1}, x_t; \boldsymbol{\theta}) &= \frac{f(x_{t-1}, x_t; \boldsymbol{\theta}) \cdot \gamma_t(x_t)}{F_t^\gamma(x_{t-1}; \boldsymbol{\theta})}, \qquad\qquad 1 < t \leq T,
\end{aligned}
$$

where $H^\gamma(\boldsymbol{\theta}) = \int h(x_1; \boldsymbol{\theta}) \gamma_1(x_1) dx_1$ and $F_t^\gamma(x_{t-1}; \boldsymbol{\theta}) = \int f(x_{t-1}, x_t; \boldsymbol{\theta}) \gamma_t(x_t) dx_t$ are normalization terms for the probability densities h^γ and f_t^γ, respectively. To ensure that the parameter likelihood estimate $\hat{p}(\boldsymbol{y} \mid \boldsymbol{\theta})$ remains unbiased under the twisted model, we define the twisted state-dependent likelihoods $g_1^\gamma, \ldots, g_T^\gamma$ as functions that satisfy:

$$
\begin{aligned}
\hat{p}(\boldsymbol{x}, \boldsymbol{y} \mid \boldsymbol{\theta}) &= h^\gamma(x_1; \boldsymbol{\theta}) \cdot \prod_{t=2}^T f_t^\gamma(x_{t-1}, x_t; \boldsymbol{\theta}) \cdot \prod_{t=1}^T g_t^\gamma(x_t, y_t; \boldsymbol{\theta}) \\
h(x_1; \boldsymbol{\theta}) \cdot \prod_{t=2}^T f(x_{t-1}, x_t; \boldsymbol{\theta}) \cdot \prod_{t=1}^T g(x_t, y_t; \boldsymbol{\theta}) &= \frac{h(x_1; \boldsymbol{\theta}) \gamma_1(x_1)}{H^\gamma(\boldsymbol{\theta})} \cdot \prod_{t=2}^T \frac{f(x_{t-1}, x_t; \boldsymbol{\theta}) \gamma_t(x_t; \boldsymbol{\theta})}{F_t^\gamma(x_{t-1}; \boldsymbol{\theta})} \cdot \prod_{t=1}^T g_t^\gamma(x_t, y_t; \boldsymbol{\theta}) \\
\prod_{t=1}^T g(x_t, y_t; \boldsymbol{\theta}) &= \frac{\gamma_1(x_1)}{H^\gamma(\boldsymbol{\theta})} \cdot \prod_{t=2}^T \frac{\gamma_t(x_t)}{F_t^\gamma(x_{t-1}; \boldsymbol{\theta})} \cdot \prod_{t=1}^T g_t^\gamma(x_t, y_t; \boldsymbol{\theta}).
\end{aligned}
$$

This equality can be maintained if we define $g_1^\gamma, \ldots, g_T^\gamma$ as follows,

$$\begin{aligned} g_1^\gamma(x_1, y_1; \boldsymbol{\theta}) &= \frac{H^\gamma(\boldsymbol{\theta}) \cdot g(x_1, y_1; \boldsymbol{\theta}) \cdot F_2^\gamma(x_1; \boldsymbol{\theta})}{\gamma_1(x_1)}, \\ g_t^\gamma(x_t, y_t; \boldsymbol{\theta}) &= \frac{g(x_t, y_t; \boldsymbol{\theta}) \cdot F_{t+1}^\gamma(x_t; \boldsymbol{\theta})}{\gamma_t(x_t)}, \qquad 1 < t < T, \\ g_T^\gamma(x_T, y_T; \boldsymbol{\theta}) &= \frac{g(x_T, y_T; \boldsymbol{\theta})}{\gamma_T(x_T)}. \end{aligned}$$

Thus, the parameter likelihood estimate of the twisted model is

$$\hat{p}^\gamma(\boldsymbol{y} \mid \boldsymbol{\theta}) = \prod_{t=1}^{T} \left(\frac{1}{S} \sum_{s=1}^{S} g_t^\gamma(x_t^s, y_t; \boldsymbol{\theta}) \right).$$

The BPF is simply a degenerate case of twisted SMC in which $\gamma_t = 1$ for all t.

B.3 Determining the Optimal Policy γ^*

The variance of the estimate $\hat{p}^\gamma$ comes from the state-dependent likelihood g. Thus, to minimize the variance, we would like g_t^γ to be as uniform as possible with respect to x_t. Let the optimal policy be denoted γ^*. It follows that

$$\begin{aligned} \gamma_T^*(x_T) &= g(x_T, y_T; \boldsymbol{\theta}), \\ \gamma_t^*(x_t) &= g(x_t, y_t; \boldsymbol{\theta}) \cdot F_{t+1}^{\gamma^*}(x_t; \boldsymbol{\theta}), \qquad 1 \le t < T. \end{aligned}$$

Under γ^*, the likelihood estimate $\hat{p}^{\gamma^*}(\boldsymbol{y} \mid \boldsymbol{\theta}) = H^{\gamma^*} = p(\boldsymbol{y} \mid \boldsymbol{\theta})$ has zero variance. However, it may be infeasible for us to use γ^* in many cases, because the BPF algorithm requires us to sample x_t from $f_t^{\gamma^*}$ for all t. For example, under γ^*, we would have

$$f_T^{\gamma^*}(x_{T-1}, x_T; \boldsymbol{\theta}) \propto f(x_{T-1}, x_T; \boldsymbol{\theta}) \cdot \gamma_T^*(x_T) = f(x_{T-1}, x_T; \boldsymbol{\theta}) \cdot g(x_T, y_T; \boldsymbol{\theta}),$$

which may be impossible to directly sample from if f and g form an intractable posterior (e.g. if f is Gaussian and g is binomial). In such a case, we must choose a suboptimal policy γ.

B.4 Choosing a Policy γ for the Neuroscience Application

Recall the point-process state-space model (Equation 4), in which we have

$$\begin{aligned} h(x_1; \boldsymbol{\theta}) &= \mathcal{N}(x_1 \mid x_0 + \mu, \psi_0), \\ f(x_{t-1}, x_t; \boldsymbol{\theta}) &= \mathcal{N}(x_t \mid x_{t-1}, \psi), \\ g(x_t, y_t) &= \text{Binomial}\left(M \cdot R, \frac{\exp x_t}{1 + \exp x_t} \right), \end{aligned}$$

where we define the parameters $\boldsymbol{\theta} = \{\mu, \log \psi\}$ and x_0, ψ_0, M, R are supplied constants.

Here, we can show that $F_{t+1}^{\gamma^*}(x_t; \boldsymbol{\theta}) = \int f(x_t, x_{t+1}; \boldsymbol{\theta}) \gamma_{t+1}^*(x_{t+1}) dx_{t+1}$ must be log-concave in x_t. This further implies that for all t, $\gamma_t^*(x_t) = g(x_t, y_t) \cdot F_{t+1}^{\gamma^*}(x_t; \boldsymbol{\theta})$ is a log-concave function of x_t since the product of two log-concave functions is log-concave. Hence, we have shown that the optimal policy $\gamma^* = \{\gamma_1^*, \ldots, \gamma_T^*\}$ is a series of log-concave functions. This justifies the approximation of each $\gamma_t^*(x_t)$ with a Gaussian function,

$$\gamma_t(x_t) = \exp(-a_t x_t^2 - b_t x_t - c_t), \quad (a_t, b_t, c_t) \in \mathbb{R}^3,$$

and thus, $f_t^\gamma(x_{t-1}, x_t; \boldsymbol{\theta}) \propto f(x_{t-1}, x_t; \boldsymbol{\theta}) \cdot \gamma_t(x_t)$ is also a Gaussian density that is easy to sample from when running the BPF algorithm.

We want to find the values of (a_t, b_t, c_t) that enforce $\gamma_t \approx \gamma_t^*$ for all t. One simple way to accomplish this goal is to find the (a_t, b_t, c_t) that minimizes the least-squares difference between γ_t and γ_t^* in log-space. That is, given a set of samples $\{x_t^1, \ldots, x_t^S\}$ for the random variable x_t, we solve for:

$$\begin{aligned}(a_t, b_t, c_t) &= \arg\min_{(a_t,b_t,c_t)\in\mathbb{R}^3} \sum_{s=1}^{S} \left[\log \gamma_t(x_t^s) - \log \gamma_t^*(x_t^s)\right]^2 \\ &= \arg\min_{(a_t,b_t,c_t)\in\mathbb{R}^3} \sum_{s=1}^{S} \left[-(a_t(x_t^s)^2 + b_t(x_t^s) + c_t) - \log \gamma_t^*(x_t^s)\right]^2.\end{aligned}$$

Also note that in a slight abuse of notation, we redefine for all $t < T$,

$$\gamma_t^*(x_t) = g(x_t, y_t) \cdot F_{t+1}^{\gamma}(x_t; \boldsymbol{\theta}),$$

because when performing approximate backwards recursion, it is not possible to analytically solve for the intractable integral $F_{t+1}^{\gamma^*}(x_t; \boldsymbol{\theta})$.

In the aforementioned least-squares optimization problem, there is one additional constraint that we must take into account. Recall that $f_t^{\gamma}(x_{t-1}, x_t; \boldsymbol{\theta}) \propto f(x_{t-1}, x_t; \boldsymbol{\theta}) \cdot \gamma_t(x_t)$ is a Gaussian pdf that we sample from. Therefore, we must ensure that the variance of this distribution is positive, which places a constraint on γ_t and more specifically, the domain of (a_t, b_t, c_t). Using properties of Gaussians, we can perform algebraic manipulation to work out the following parameterizations of h^{γ} and f_t^{γ}:

$$\begin{aligned}h^{\gamma}(x_1; \boldsymbol{\theta}) &= \mathcal{N}\left(x_1 \,\middle|\, \frac{\psi_0^{-1} \cdot (x_0 + \mu) - b_1}{\psi_0^{-1} + 2a_1}, \frac{1}{\psi_0^{-1} + 2a_1}\right), \\ f_t^{\gamma}(x_{t-1}, x_t; \boldsymbol{\theta}) &= \mathcal{N}\left(x_t \,\middle|\, \frac{\psi^{-1} \cdot x_{t-1} - b_t}{\psi^{-1} + 2a_t}, \frac{1}{\psi^{-1} + 2a_t}\right), \qquad 1 < t \leq T.\end{aligned}$$

The corresponding normalizing terms for these densities are

$$\begin{aligned}H^{\gamma}(\boldsymbol{\theta}) &= \frac{1}{\sqrt{1 + 2a_1\psi_0}} \exp\left(\frac{\psi_0^{-1} \cdot (x_0 + \mu) - (b_1)^2}{2(\psi_0^{-1} + 2a_1)} - \frac{(x_0 + \mu)^2}{2\psi_0} - c_1\right), \\ F_t^{\gamma}(x_{t-1}; \boldsymbol{\theta}) &= \frac{1}{\sqrt{1 + 2a_t\psi}} \exp\left(\frac{\psi^{-1} \cdot x_{t-1} - (b_t)^2}{2(\psi^{-1} + 2a_t)} - \frac{x_{t-1}^2}{2\psi} - c_t\right), \qquad 1 < t \leq T.\end{aligned}$$

Thus, to obtain (a_t, b_t, c_t) and consequently γ_t for all t, we solve the aforementioned least-squares minimization problem subject to the following constraints:

$$a_1 > -\frac{1}{2\psi_0}, \qquad a_t > -\frac{1}{2\psi}, \quad 1 < t \leq T.$$

B.5 Full cSMC Algorithm

The full controlled sequential Monte Carlo algorithm iterates on twisted SMC for L iterations, building a series of policies $\gamma^{\langle 1 \rangle}, \gamma^{\langle 2 \rangle}, \ldots, \gamma^{\langle L \rangle}$ over time. Given two policies Γ' and γ, we can define

$$\begin{aligned}h^{\Gamma' \cdot \gamma}(x_1) &\propto h^{\Gamma'}(x_1)\gamma_1(x_1) = h(x_1; \boldsymbol{\theta}) \cdot \Gamma_1'(x_1) \cdot \gamma_1(x_1), \\ f_t^{\Gamma' \cdot \gamma}(x_{t-1}, x_t; \boldsymbol{\theta}) &\propto f_t^{\Gamma'}(x_{t-1}, x_t; \boldsymbol{\theta}) \cdot \gamma_t(x_t) = f(x_{t-1}, x_t; \boldsymbol{\theta}) \cdot \Gamma_t'(x_t) \cdot \gamma_t(x_t), \qquad 1 < t \leq T.\end{aligned}$$

We can see from these relationships that twisting the original model using Γ' and then twisting the new model using γ has the same effect as twisting the original model using a cumulative policy Γ where each $\Gamma_t(x_t) = \Gamma_t'(x_t) \cdot \gamma_t(x_t)$. We state the full cSMC algorithm in Algorithm 3.

Algorithm 3 `ControlledSMC`($\boldsymbol{y}$, g, μ, ψ, x_0, ψ_0, L)

Define $f(x_{t-1}, x_t; \boldsymbol{\theta}) = \mathcal{N}(x_t \mid x_{t-1}, \psi)$ and $h(x_1; \boldsymbol{\theta}) = \mathcal{N}(x_1 \mid x_0 + \mu, \psi_0)$.
Define parameters $\boldsymbol{\theta} = \{\mu, \log \psi\}$.
Collect particles $\{x_1^s\}_{s=1}^S, \ldots, \{x_T^s\}_{s=1}^S$ from `BootstrapParticleFilter`($\boldsymbol{y}$, $\boldsymbol{\theta}$, f, g, h).
Initialize $\Gamma' = \{\Gamma'_1, \ldots, \Gamma'_T\}$ where $\Gamma'_t(x_t) = 1$ for all $t = 1, \ldots, T$.
Initialize $g_t^{\Gamma'}(x_t, y_t) = g(x_t, y_t)$ for all $t = 1, \ldots, T$.
Initialize $a_t^{\langle 0 \rangle} = 0, b_t^{\langle 0 \rangle} = 0, c_t^{\langle 0 \rangle} = 0$ for all $t = 1, \ldots, T$.
for $\ell = 1, \ldots, L$ **do**
  // Approximate backward recursion to determine policy and associated functions
  Define $\gamma_T^*(x_T) = g_T^{\Gamma'}(x_T, y_T)$.
  for $t = T, \ldots, 2$ **do**
    Solve $(a_t^{\langle \ell \rangle}, b_t^{\langle \ell \rangle}, c_t^{\langle \ell \rangle}) = \arg\min_{(a_t, b_t, c_t)} \sum_{s=1}^S \left[-(a_t(x_t^s)^2 + b_t(x_t^s) + c_t) - \log \gamma_t^*(x_t^s)\right]^2$ subject to $a_t > -1/(2\psi) - \sum_{\ell'=0}^{\ell-1} a_t^{\langle \ell' \rangle}$ using linear regression.
    Define new policy function $\gamma_t(x_t) = \exp(-a_t^{\langle \ell \rangle} x_t^2 - b_t^{\langle \ell \rangle} x_t - c_t^{\langle \ell \rangle})$.
    Define cumulative policy function $\Gamma_t(x_t) = \Gamma'_t(x_t) \cdot \gamma_t(x_t) = \exp(-A_t x_t^2 - B_t x_t - C_t)$ where $A_t = \sum_{\ell'=0}^{\ell} a_t^{\langle \ell' \rangle}$, $B_t = \sum_{\ell'=0}^{\ell} b_t^{\langle \ell' \rangle}$, and $C_t = \sum_{\ell'=0}^{\ell} c_t^{\langle \ell' \rangle}$.
    Define $f_t^\Gamma(x_{t-1}, x_t; \boldsymbol{\theta})$ and $F_t^\Gamma(x_{t-1}; \boldsymbol{\theta})$.
    if $t = T$ **then**
      Define $g_T^\Gamma(x_T, y_T) = g(x_T, y_T)/\Gamma_T(x_T)$.
    else
      Define $g_t^\Gamma(x_t, y_t) = g(x_t, y_t) \cdot F_{t+1}^\Gamma(x_t; \boldsymbol{\theta})/\Gamma_t(x_t)$.
    end if
    Define $\gamma_{t-1}^*(x_{t-1}) = g_{t-1}^{\Gamma'}(x_{t-1}, y_{t-1}) \cdot F_t^\Gamma(x_{t-1}; \boldsymbol{\theta})/F_t^{\Gamma'}(x_{t-1}; \boldsymbol{\theta})$.
  end for
  Solve $(a_1^{\langle \ell \rangle}, b_1^{\langle \ell \rangle}, c_1^{\langle \ell \rangle}) = \arg\min_{(a_1, b_1, c_1)} \sum_{s=1}^S \left[-(a_1(x_1^s)^2 + b_1(x_1^s) + c_1) - \log \gamma_1^*(x_1^s)\right]^2$ subject to $a_1 > -1/(2\psi_0) - \sum_{\ell'=0}^{\ell-1} a_1^{\langle \ell' \rangle}$ using linear regression.
  Define new policy function $\gamma_1(x_1) = \exp(-a_1^{\langle \ell \rangle} x_1^2 - b_1^{\langle \ell \rangle} x_1 - c_1^{\langle \ell \rangle})$.
  Define cumulative policy function $\Gamma_1(x_1) = \Gamma'_1(x_1) \cdot \gamma_1(x_1) = \exp(-A_1 x_t^2 - B_1 x_t - C_1)$ where $A_1 = \sum_{\ell'=0}^{\ell} a_1^{\langle \ell' \rangle}$, $B_1 = \sum_{\ell'=0}^{\ell} b_1^{\langle \ell' \rangle}$, and $C_1 = \sum_{\ell'=0}^{\ell} c_1^{\langle \ell' \rangle}$.
  Define Γ-twisted initial prior $h^\Gamma(x_1; \boldsymbol{\theta})$ and $H^\Gamma(\boldsymbol{\theta})$.
  Define $g_1^\Gamma(x_1, y_1) = H^\Gamma(\boldsymbol{\theta}) \cdot g(x_1, y_1) \cdot F_2^\Gamma(x_1; \boldsymbol{\theta})/\Gamma_1(x_1)$.
  // Forward bootstrap particle filter to sample particles and compute weights
  for $s = 1, \ldots, S$ **do**
    Sample $x_1^s \sim h^\Gamma(x_1)$ and weight $w_1^s = g_1^\Gamma(x_1^s, y_1)$.
  end for
  Normalize $\{w_1^s\}_{s=1}^S = \{w_1^s\}_{s=1}^S / \sum_{s=1}^S w_1^s$.
  for $t = 2, \ldots, T$ **do**
    for $s = 1, \ldots, S$ **do**
      Resample ancestor index $a \sim \text{Categorical}(w_{t-1}^1, \ldots, w_{t-1}^S)$.
      Sample $x_t^s \sim f_t^\Gamma(x_{t-1}^a, x_t; \boldsymbol{\theta})$ and weight $w_t^s = g_t^\Gamma(x_t^s, y_t)$.
    end for
    Normalize $\{w_t^s\}_{s=1}^S = \{w_t^s\}_{s=1}^S / \sum_{s=1}^S w_t^s$.
  end for
  Update $\Gamma' = \Gamma$.
end for
return Likelihood estimate $\hat{p}^\Gamma(\boldsymbol{y} \mid \boldsymbol{\theta})$.

C MULTIPLE GIBBS SAMPLES WITH OPTIMAL CO-OCCURENCE MATRIX

This section extends the method of selecting clusters detailed in Section 4.1. After running the DPnSSM inference algorithm (Algorithm 1), we construct co-occurrence matrices $\mathbf{\Omega}^{\langle i \rangle}$ for $i = 1, \ldots, I$ (Equation 10). Then, we select the optimal Gibbs sample i^*. If there are $J > 1$ Gibbs samples $i_1, \ldots, i_J$ such that $\mathbf{\Omega}^{\langle i_j \rangle} = \mathbf{\Omega}^{\langle i^* \rangle}$ for $j = 1, \ldots, J$, our final cluster parameters $\mathbf{\Theta}^*$ can be redefined as the average among the corresponding parameter samples,

$$\mathbf{\Theta}^* = \frac{1}{J} \sum_{j=1}^{J} \mathbf{\Theta}^{\langle i_j \rangle} \approx \mathbb{E}[\mathbf{\Theta} \mid Z = Z^*]. \tag{15}$$

This averaging must be preceded by a permutation of each set of $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(K)} \in \mathbf{\Theta}^{\langle i_j^* \rangle}$ to fix any potential label switching.

D ROBUSTNESS OF MODEL TO STIMULUS MISSPECIFICATION

We extend the results of Section 4.2 by testing the robustness of the model under cases in which there is a mismatch between the true stimulus onset and the model's specification of the stimulus onset. In particular, we first examine the case in which the model overpredicts the true stimulus onset. Figure 8 presents heatmaps of the mean co-occurrence matrix $\mathbf{\Omega}$ in cases in which the model's anticipation of the stimulus falls 40 ms, 80 ms, and 160 ms behind the true onset. Table 4 lists the parameters chosen in the final clustering of the data in these three cases. In all experiments, we use the same data generation process as detailed in Section 4.2.1 and the same modeling process as detailed in Section 4.2.2.

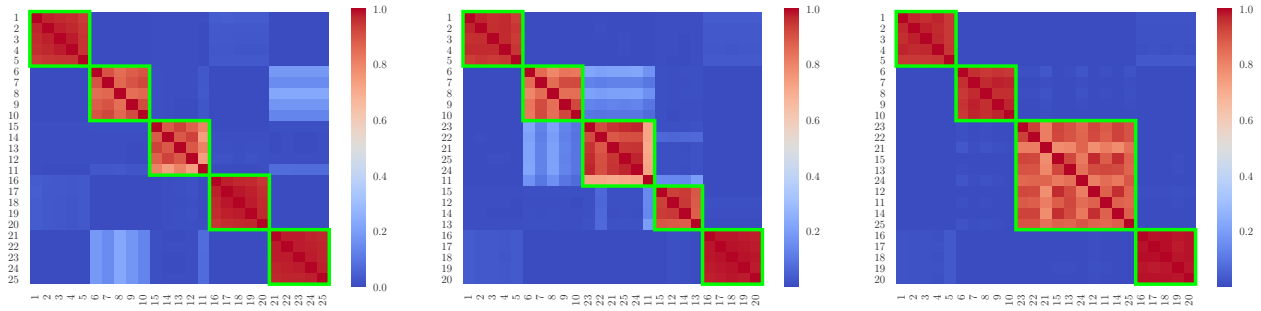

Figure 8: Results from running the DPnSSM inference algorithm in cases in which the model specifies the stimulus as occurring (left) 40 ms, (center) 80 ms, and (right) 160 ms *after* the true onset. At 40 ms, the ground-truth clustering can be recovered, but this ability decays as the time difference increases.

Table 4: Final cluster parameters for model stimulus delay. As expected, the absolute value of $\mu^{*(k)}$ decreases for all k as time mismatch increases.

	40 ms model delay			80 ms model delay			160 ms model delay		
k	$\mu^{*(k)}$	$\log \psi^{*(k)}$	# of Neurons	$\mu^{*(k)}$	$\log \psi^{*(k)}$	# of Neurons	$\mu^{*(k)}$	$\log \psi^{*(k)}$	# of Neurons
1	+0.86	−10.86	5	+0.80	−10.95	5	+0.65	−10.79	5
2	−0.88	−12.12	5	−0.84	−12.45	5	−0.76	−12.48	5
3	+0.02	−10.31	5	−0.62	−6.23	6	+0.01	−9.36	10
4	+0.94	−5.53	5	+0.05	−10.77	4	+0.57	−5.47	5
5	−0.87	−5.91	5	+0.87	−5.52	5			

Next, we examine the case in which the model underpredicts the true stimulus onset – by 10 ms, 20 ms, and 40 ms. This set of results is less robust than the previous one. Heatmaps are given in Figure 9, while parameters are given in Table 5.

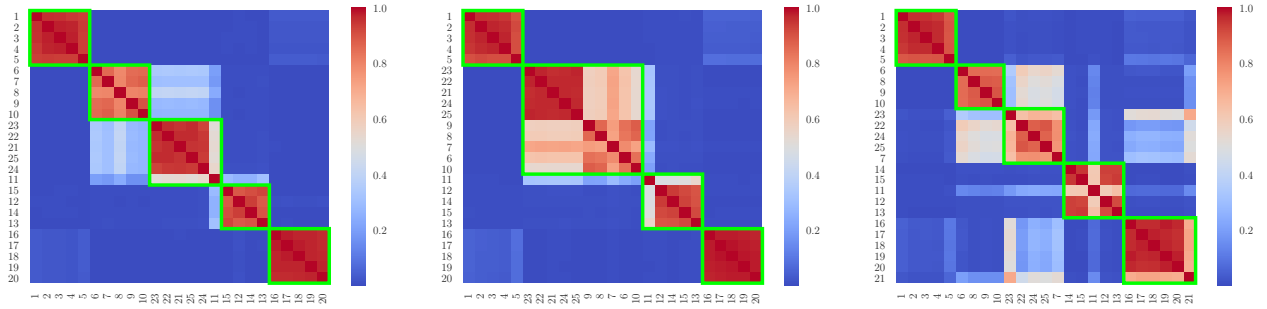

Figure 9: Results from running the DPnSSM inference algorithm in cases in which the model specifies the stimulus as occurring (left) 10 ms, (center) 20 ms, and (right) 40 ms *before* the true onset. At 10 ms, the ground-truth clustering can almost be fully recovered, but this ability significantly decays as the time difference increases. For a true stimulus delay of 40 ms, there even exists one cluster containing both excited and inhibited neurons.

Table 5: Final cluster parameters for true stimulus delay.

	10 ms stimulus delay			20 ms stimulus delay			40 ms stimulus delay		
k	$\mu^{*(k)}$	$\log \psi^{*(k)}$	# of Neurons	$\mu^{*(k)}$	$\log \psi^{*(k)}$	# of Neurons	$\mu^{*(k)}$	$\log \psi^{*(k)}$	# of Neurons
1	$+0.93$	-10.45	5	$+0.91$	-10.37	5	$+0.79$	-8.71	5
2	-0.90	-12.47	4	-0.63	-6.52	10	-0.89	-11.5	4
3	-0.66	-6.08	6	$+0.03$	-10.02	5	-0.34	-5.93	5
4	$+0.07$	-10.96	5	$+0.67$	-5.31	5	$+0.02$	-10.41	5
5	$+0.82$	-5.44	5				$+0.34$	-4.96	6

E ADDITIONAL RASTER PLOTS FOR CUE DATA OVER TIME

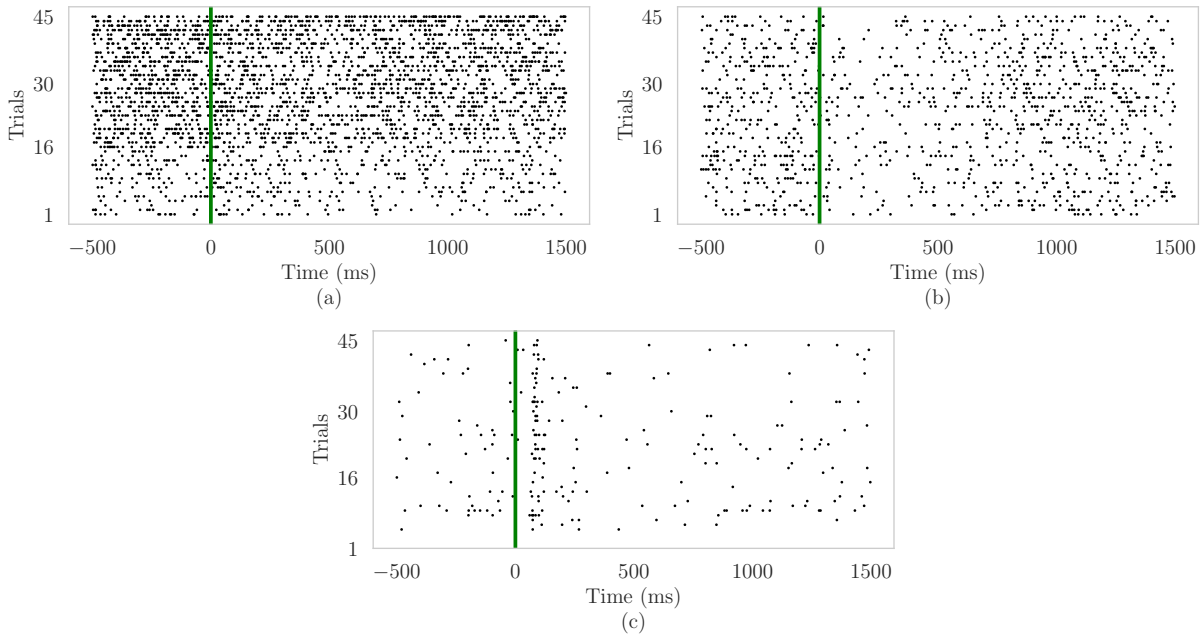

Figure 10: Overlaid raster plots of additional neuronal clusters found in Section 4.3.2 for (a) cluster $k = 2$ with eight neurons and slightly inhibited/sustained responses, (b) cluster $k = 4$ with four neurons and more strongly inhibited/variable responses, and (c) cluster $k = 2$ with a single neuron and a delayed excited effect. A black dot at (τ, r) indicates a spike from one of the neurons in the corresponding cluster at time τ during trial r. The vertical green line indicates cue onset.

F DETAILS OF CSMC VS. BPF EXPERIMENT

This section describes the experimental setup of Section 4.4 that is used to produce Figure 7. Computational cost is fixed at approximately 35 milliseconds per likelihood evaluation for either method. For the bootstrap particle filter (BPF), we use $S = 1024$ particles. For controlled sequential Monte Carlo (cSMC), we use $S = 64$ particles and $L = 3$ iterations. All parameter log-likelihood evaluations are performed on a representative real neuron's cue data over time $\boldsymbol{y}$, as described in Section 4.3.1. For each particle filtering method, let $v(\mu, \psi) = \mathbb{V}\mathrm{ar}[\log p(\boldsymbol{y} \mid \boldsymbol{\theta})]$, where $\boldsymbol{\theta} = [\mu, \log \psi]^\top$. Figure 7 plots empirical estimates of $\hat{v}$ of v over different values of (μ, ψ) for the two methods. For each empirical variance estimate, we use 500 estimates of $\log p(\boldsymbol{y} \mid \boldsymbol{\theta})$. As $\log \psi$ decreases, cSMC performs substantially better than BPF, especially for extreme values of μ.