Appendix for Bayesian Clustering of Neural Spiking Activity Using a mixDPFA

A Appendix

A.1 MCMC updates

The posteriors are sampled by a Gibbs sampler. In each iteration, the sampling scheme has two main stages: 1) to sample the model parameters assuming known labels and 2) to sample the cluster indices given model parameters. Before moving into the second stage, the first stage is repeated several times. The sampling in the first stage is conducted without considering constraints for $\mu_t^{(j)}$ and $\boldsymbol{x}_t^{(j)}$ at first, and then we project the samples onto the constraint space for $\sum_{t=1}^T \mu_t^{(j)} = 0$ and $\sum_{t=1}^T \boldsymbol{x}_t^{(j)} = \mathbf{0}$ as used in (Sen et al., 2018).

Update $\boldsymbol{x}_t^{(j)}$ and $\mu_t^{(j)}$ The priors for initial population baseline and latent factor are:

$$\mu_1^{(j)} \sim \mathcal{N}(0, 1),$$
$$\boldsymbol{x}_1^{(j)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_p).$$

Assume there are $n_j = \#\{i : z_i = j\}$ neurons belong to the j-th cluster, and denote the spike counts and firing rates of these neurons at time t as $\widetilde{\boldsymbol{y}}_t^{(j)} = vec(\{y_{it}|z_i = j\}) \in \mathbb{Z}_{\geq 0}^{n_j}$ and $\widetilde{\boldsymbol{\lambda}}_t^{(j)} = vec(\{\lambda_{it}|z_i = j\}) \in \mathbb{R}^{n_j}$, where $\mathbb{Z}_{\geq 0}^{n_j}$ denotes a n_j-dimensional vector with each element being non-negative integers. The corresponding loading and baseline for these n_j neurons is $\boldsymbol{C}^{(j)} \in \mathbb{R}^{n_j \times p}$ and $\boldsymbol{\Delta}_j = vec(\{\delta_i|z_i = j\}) \in \mathbb{R}^{n_j}$, such that $\log \widetilde{\boldsymbol{\lambda}}_t^{(j)} = \boldsymbol{\Delta}_j + \mu_t^{(j)} \mathbf{1}_{n_j} + \boldsymbol{C}^{(j)} \boldsymbol{x}_t^{(j)} = \boldsymbol{\Delta}_j + \left(\mathbf{1}_{n_j}, \boldsymbol{C}^{(j)}\right) \left(\mu_t^{(j)}, \boldsymbol{x}_t'^{(j)}\right)'$. Denote $\widetilde{\boldsymbol{C}}^{(j)} = \left(\mathbf{1}_{n_j}, \boldsymbol{C}^{(j)}\right)$, $\widetilde{\boldsymbol{x}}_t^{(j)} = \left(\mu_t^{(j)}, \boldsymbol{x}_t'^{(j)}\right)'$, $\widetilde{\boldsymbol{x}}^{(j)} = \left(\widetilde{\boldsymbol{x}}_1'^{(j)}, \ldots, \widetilde{\boldsymbol{x}}_T'^{(j)}\right)'$, $\widetilde{\boldsymbol{A}}^{(j)} = diag(h^{(j)}, \boldsymbol{A}^{(j)})$, $\widetilde{\boldsymbol{b}}^{(j)} = \left(g^{(j)}, \boldsymbol{b}'^{(j)}\right)'$ and $\widetilde{\boldsymbol{Q}}^{(j)} = diag(\sigma^{2(j)}, \boldsymbol{Q}^{(j)})$. The full conditional distribution $P(\widetilde{\boldsymbol{x}}^{(j)}|\ldots) = P(\widetilde{\boldsymbol{x}}^{(j)}|\{\widetilde{\boldsymbol{y}}_t^{(j)}\}_{t=1}^T, \boldsymbol{\Delta}_j, \widetilde{\boldsymbol{C}}^{(j)}, \widetilde{\boldsymbol{A}}^{(j)}, \widetilde{\boldsymbol{b}}^{(j)}, \widetilde{\boldsymbol{Q}}^{(j)})$ is approximated by a global Laplace approximation, i.e.,

$$P(\widetilde{\boldsymbol{x}}^{(j)}|\ldots) \approx \mathcal{N}_{(p+1)T}(\widetilde{\boldsymbol{x}}^{(j)}|\boldsymbol{\mu}_{\widetilde{\boldsymbol{x}}^{(j)}}, \boldsymbol{\Sigma}_{\widetilde{\boldsymbol{x}}^{(j)}}),$$
$$\boldsymbol{\mu}_{\widetilde{\boldsymbol{x}}^{(j)}} = \arg\max_{\widetilde{\boldsymbol{x}}^{(j)}} P(\widetilde{\boldsymbol{x}}^{(j)}|\ldots),$$
$$\boldsymbol{\Sigma}_{\widetilde{\boldsymbol{x}}^{(j)}}) = -(\nabla\nabla \log P(\widetilde{\boldsymbol{x}}^{(j)}|\ldots)|_{\widetilde{\boldsymbol{x}}^{(j)} = \boldsymbol{\mu}_{\widetilde{\boldsymbol{x}}^{(j)}}})^{-1}.$$

Then, taking the logarithm of the full conditional distribution, i.e., $\ell = \ell(\widetilde{\boldsymbol{x}}^{(j)}) = \log P(\widetilde{\boldsymbol{x}}^{(j)}|\ldots)$, we have

$$\ell = \text{const} + \sum_{t=1}^T \left(\widetilde{\boldsymbol{y}}_t'^{(j)}(\boldsymbol{\Delta}_j + \widetilde{\boldsymbol{C}}^{(j)}\widetilde{\boldsymbol{x}}_t^{(j)}) - \mathbf{1}_{n_j}' \widetilde{\boldsymbol{\lambda}}_t\right) - \frac{1}{2}(\widetilde{\boldsymbol{x}}_1^{(j)} - \widetilde{\boldsymbol{x}}_0'^{(j)})[\widetilde{\boldsymbol{Q}}_0^{(j)}]^{-1}(\widetilde{\boldsymbol{x}}_1^{(j)} - \widetilde{\boldsymbol{x}}_0^{(j)})$$
$$- \sum_{t=2}^T \frac{1}{2}(\widetilde{\boldsymbol{x}}_t^{(j)} - \widetilde{\boldsymbol{A}}^{(j)}\widetilde{\boldsymbol{x}}_{t-1}^{(j)} - \widetilde{\boldsymbol{b}}'^{(j)})[\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}(\widetilde{\boldsymbol{x}}_t^{(j)} - \widetilde{\boldsymbol{A}}^{(j)}\widetilde{\boldsymbol{x}}_{t-1}^{(j)} - \widetilde{\boldsymbol{b}}^{(j)}).$$

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Here, $\ell(\widetilde{\boldsymbol{x}}^{(j)})$ is concave and unimodal. By the Markovian assumption for latent state vectors, the Hessian matrix is tri-block diagonal. We can thus compute Newton updates to get $\boldsymbol{\mu}_{\widetilde{\boldsymbol{x}}^{(j)}}$ for the Laplace approximation in $\mathcal{O}(T)$ (Paninski et al., 2010), similar to the E-step for the PLDS (Macke et al., 2011).

The gradient $\nabla\ell$ and the Hessian $\nabla\nabla\ell$ are provided as follows. For $t = 2, \ldots, T-1$, the gradient of $\ell(\widetilde{\boldsymbol{x}}^{(j)})$ is:

$$
\begin{aligned}
\nabla\ell &= \left[\left(\frac{\partial\ell}{\partial\widetilde{\boldsymbol{x}}_1^{(j)}}\right)', \ldots, \left(\frac{\partial\ell}{\partial\widetilde{\boldsymbol{x}}_T^{(j)}}\right)'\right]', \\
\frac{\partial\ell}{\partial\widetilde{\boldsymbol{x}}_1^{(j)}} &= \widetilde{\boldsymbol{C}}'^{(j)}(\widetilde{\boldsymbol{y}}_1^{(j)} - \widetilde{\boldsymbol{\lambda}}_1) - [\widetilde{\boldsymbol{Q}}_0^{(j)}]^{-1}(\widetilde{\boldsymbol{x}}_1^{(j)} - \widetilde{\boldsymbol{x}}_0^{(j)}) + \widetilde{\boldsymbol{A}}'^{(j)}[\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}(\widetilde{\boldsymbol{x}}_2^{(j)} - \widetilde{\boldsymbol{A}}^{(j)}\widetilde{\boldsymbol{x}}_1^{(j)} - \widetilde{\boldsymbol{b}}^{(j)}), \\
\frac{\partial\ell}{\partial\widetilde{\boldsymbol{x}}_t^{(j)}} &= \widetilde{\boldsymbol{C}}'^{(j)}(\widetilde{\boldsymbol{y}}_t^{(j)} - \widetilde{\boldsymbol{\lambda}}_t) - [\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}(\widetilde{\boldsymbol{x}}_t^{(j)} - \widetilde{\boldsymbol{A}}^{(j)}\widetilde{\boldsymbol{x}}_{t-1}^{(j)} - \widetilde{\boldsymbol{b}}^{(j)}) \\
&\quad + \widetilde{\boldsymbol{A}}'^{(j)}[\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}(\widetilde{\boldsymbol{x}}_{t+1}^{(j)} - \widetilde{\boldsymbol{A}}^{(j)}\widetilde{\boldsymbol{x}}_t^{(j)} - \widetilde{\boldsymbol{b}}^{(j)}), \\
\frac{\partial\ell}{\partial\widetilde{\boldsymbol{x}}_T^{(j)}} &= \widetilde{\boldsymbol{C}}'^{(j)}(\widetilde{\boldsymbol{y}}_t^{(j)} - \widetilde{\boldsymbol{\lambda}}_T) - [\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}(\widetilde{\boldsymbol{x}}_T^{(j)} - \widetilde{\boldsymbol{A}}^{(j)}\widetilde{\boldsymbol{x}}_{T-1}^{(j)} - \widetilde{\boldsymbol{b}}^{(j)}).
\end{aligned}
$$

And the Hessian matrix is:

$$
\nabla\nabla\ell = \begin{pmatrix}
\frac{\partial^2\ell}{\partial\widetilde{\boldsymbol{x}}_1^{(j)}\partial\widetilde{\boldsymbol{x}}_1'^{(j)}} & \widetilde{\boldsymbol{A}}'^{(j)}[\widetilde{\boldsymbol{Q}}^{(j)}]^{-1} & \mathbf{0} & \cdots & \mathbf{0} \\
[\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}\widetilde{\boldsymbol{A}}^{(j)} & \frac{\partial^2\ell}{\partial\widetilde{\boldsymbol{x}}_2^{(j)}\partial\widetilde{\boldsymbol{x}}_2'^{(j)}} & \widetilde{\boldsymbol{A}}'^{(j)}[\widetilde{\boldsymbol{Q}}^{(j)}]^{-1} & \cdots & \vdots \\
\mathbf{0} & [\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}\widetilde{\boldsymbol{A}}^{(j)} & \frac{\partial^2\ell}{\partial\widetilde{\boldsymbol{x}}_3^{(j)}\partial\widetilde{\boldsymbol{x}}_3'^{(j)}} & \cdots & \vdots \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\mathbf{0} & \cdots & \cdots & \cdots & \frac{\partial^2\ell}{\partial\widetilde{\boldsymbol{x}}_T^{(j)}\partial\widetilde{\boldsymbol{x}}_T'^{(j)}}
\end{pmatrix},
$$

$$
\begin{aligned}
\frac{\partial^2\ell}{\partial\widetilde{\boldsymbol{x}}_1^{(j)}\partial\widetilde{\boldsymbol{x}}_1'^{(j)}} &= -\widetilde{\boldsymbol{C}}'^{(j)}diag(\widetilde{\boldsymbol{\lambda}}_1)\widetilde{\boldsymbol{C}}^{(j)} - [\widetilde{\boldsymbol{Q}}_0^{(j)}]^{-1} - \widetilde{\boldsymbol{A}}'^{(j)}[\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}\widetilde{\boldsymbol{A}}^{(j)}, \\
\frac{\partial^2\ell}{\partial\widetilde{\boldsymbol{x}}_t^{(j)}\partial\widetilde{\boldsymbol{x}}_t'^{(j)}} &= -\widetilde{\boldsymbol{C}}'^{(j)}diag(\widetilde{\boldsymbol{\lambda}}_t)\widetilde{\boldsymbol{C}}^{(j)} - [\widetilde{\boldsymbol{Q}}^{(j)}]^{-1} - \widetilde{\boldsymbol{A}}'^{(j)}[\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}\widetilde{\boldsymbol{A}}^{(j)}, \\
\frac{\partial^2\ell}{\partial\widetilde{\boldsymbol{x}}_T^{(j)}\partial\widetilde{\boldsymbol{x}}_T'^{(j)}} &= -\widetilde{\boldsymbol{C}}'^{(j)}diag(\widetilde{\boldsymbol{\lambda}}_T)\widetilde{\boldsymbol{C}}^{(j)} - [\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}.
\end{aligned}
$$

Although we can conduct the Newton update efficiently in $\mathcal{O}(T)$, bad initial values may slow down the convergence. To facilitate convergence, we initialize the Newton update with a smoothing estimate by local Gaussian approximation. The forward filtering for a dynamic Poisson model has been previously described (Eden et al., 2004), and we use an additional backward pass to smooth (Rauch et al., 1965).

Let $\widetilde{\boldsymbol{x}}_{t|t-1}^{(j)} = E(\widetilde{\boldsymbol{x}}_t^{(j)}|\widetilde{\boldsymbol{y}}_1^{(j)}, \ldots, \widetilde{\boldsymbol{y}}_{t-1}^{(j)})$ and $\boldsymbol{V}_{t|t-1}^{(j)} = Var(\widetilde{\boldsymbol{x}}_t^{(j)}|\widetilde{\boldsymbol{y}}_1^{(j)}, \ldots, \widetilde{\boldsymbol{y}}_{t-1}^{(j)})$ be the mean and variance for the one-step prediction density, where $\widetilde{\boldsymbol{x}}_{t|t-1}^{(j)} = \widetilde{\boldsymbol{A}}^{(j)}\widetilde{\boldsymbol{x}}_{t-1|t-1}^{(j)} + \widetilde{\boldsymbol{b}}^{(j)}$ and $\boldsymbol{V}_{t|t-1}^{(j)} = \widetilde{\boldsymbol{A}}^{(j)}\boldsymbol{V}_{t-1|t-1}^{(j)}\widetilde{\boldsymbol{A}}'^{(j)} + \widetilde{\boldsymbol{Q}}^{(j)}$. Then, after we observe the data at time t, we can do a forward filtering step for the mean $\widetilde{\boldsymbol{x}}_{t|t}^{(j)} = E(\widetilde{\boldsymbol{x}}_t^{(j)}|\widetilde{\boldsymbol{y}}_1^{(j)}, \ldots, \widetilde{\boldsymbol{y}}_t^{(j)})$ and the variance $\boldsymbol{V}_{t|t}^{(j)} = Var(\widetilde{\boldsymbol{x}}_t^{(j)}|\widetilde{\boldsymbol{y}}_1^{(j)}, \ldots, \widetilde{\boldsymbol{y}}_t^{(j)})$, which are given by

$$
\begin{aligned}
\widetilde{\boldsymbol{x}}_{t|t}^{(j)} &= \widetilde{\boldsymbol{x}}_{t|t-1}^{(j)} + \boldsymbol{V}_{t|t-1}^{(j)}[\widetilde{\boldsymbol{C}}'^{(j)}(\widetilde{\boldsymbol{y}}_t^{(j)} - \widetilde{\boldsymbol{\lambda}}_t)]_{\widetilde{\boldsymbol{x}}_t^{(j)} = \widetilde{\boldsymbol{x}}_{t|t-1}^{(j)}}, \\
[\boldsymbol{V}_{t|t}^{(j)}]^{-1} &= [\boldsymbol{V}_{t|t-1}^{(j)}]^{-1} + [\widetilde{\boldsymbol{C}}'^{(j)}diag(\widetilde{\boldsymbol{\lambda}}_t)\widetilde{\boldsymbol{C}}^{(j)} - [\widetilde{\boldsymbol{Q}}^{(j)}]^{-1}]_{\widetilde{\boldsymbol{x}}_t^{(j)} = \widetilde{\boldsymbol{x}}_{t|t-1}^{(j)}}.
\end{aligned}
$$

Derivation of the filtering estimates can be found in (Eden et al., 2004), and we can further get the smoothing estimates directly by standard Rauch-Tung-Striebel smoother (Rauch et al., 1965).

The smoother estimates $\widetilde{\boldsymbol{x}}_{t|T}^{(j)}$ and $\boldsymbol{V}_{t|T}^{(j)}$ are updated as follows:

$$\widetilde{\boldsymbol{x}}_{t-1|T}^{(j)} = \widetilde{\boldsymbol{x}}_{t-1|t-1}^{(j)} + \boldsymbol{J}_{t-1}(\widetilde{\boldsymbol{x}}_{t|T}^{(j)} - \widetilde{\boldsymbol{x}}_{t|t-1}^{(j)}),$$
$$\boldsymbol{V}_{t-1|T}^{(j)} = \boldsymbol{V}_{t-1|t-1}^{(j)} + \boldsymbol{J}_{t-1}(\boldsymbol{V}_{t|T}^{(j)} - \boldsymbol{V}_{t|t-1}^{(j)})\boldsymbol{J}_{t-1}',$$

where $\boldsymbol{J}_{t-1} = \boldsymbol{V}_{t-1|t-1}^{(j)}\widetilde{\boldsymbol{A}}'^{(j)}[\boldsymbol{V}_{t|t-1}^{(j)}]^{-1}$.

Update δ_i and $\boldsymbol{c}_i$ We specify the prior for neuron-specific baseline δ_i as $\delta_i \sim \mathcal{N}(0,1)$ and we have assumed the loading $\boldsymbol{c}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_p)$. Then, from the matrix representation of DPFA in Equation (1), i.e., $\log \boldsymbol{\lambda}_i = \delta_i \boldsymbol{1}_T + \boldsymbol{\mu}^{(j)} + \boldsymbol{X}^{(j)}\boldsymbol{c}_i$, it is easy to see that given $\boldsymbol{\mu}^{(j)}$ and $\boldsymbol{X}^{(j)}$ are known, the update of δ_i and $\boldsymbol{c}_i$ is just a regular Bayesian Poisson regression problem. Thus, we can sample the full conditional distribution of δ_i and $\boldsymbol{c}_i$ by a Hamiltonian Monte Carlo (HMC, Duane et al. 1987) within the Gibbs sampler.

Update parameters of latent state The parameters for linear dynamics are $h^{(j)}$, $g^{(j)}$, $\sigma^{2(j)}$, $\boldsymbol{A}^{(j)}$, $\boldsymbol{b}^{(j)}$ and $\boldsymbol{Q}^{(j)}$. To make the model identifiable, we simply assume $\boldsymbol{A}^{(j)} = diag(a_1^{(j)}, \ldots, a_p^{(j)})$ and $\boldsymbol{Q}^{(j)} = diag(q_1^{(j)}, \ldots, q_p^{(j)})$. Therefore, we can update $\boldsymbol{A}^{(j)}$, $\boldsymbol{b}^{(j)}$ and $\boldsymbol{Q}^{(j)}$ for each diagonal element separately, as the update in $h^{(j)}$, $g^{(j)}$ and $\sigma^{2(j)}$. Here, we update $h^{(j)}$, $g^{(j)}$ and $\sigma^{2(j)}$ as follows.

First, we specify the priors for $\sigma^{2(j)}$ following $IG\left(\nu_0/2, \nu_0\sigma_0^2/2\right)$ and $\left(g^{(j)}, h^{(j)}\right)' \sim \mathcal{N}(\boldsymbol{\tau}_0, \sigma^{2(j)}\boldsymbol{\Lambda}_0^{-1})$, with $\nu_0 = 1$, $\sigma_0 = 0.01$, $\boldsymbol{\tau}_0 = (0,1)'$ and $\boldsymbol{\Lambda}_0 = \boldsymbol{I}_2$. Here, the "IG" denotes the inverse-gamma distribution.

Denote $\boldsymbol{\mu}_{2:T}^{(j)} = \left(\mu_2^{(j)}, \ldots, \mu_T^{(j)}\right)'$ and $\widetilde{\boldsymbol{\mu}}_{1:(T-1)}^{(j)} = \left(\boldsymbol{1}_{T-1}, \boldsymbol{\mu}_{1:(T-1)}^{(j)}\right)$, with $\boldsymbol{\mu}_{1:(T-1)}^{(j)} = \left(\mu_1^{(j)}, \ldots, \mu_{T-1}^{(j)}\right)'$. The full conditional distributions for $\sigma^{2(j)}$ and $\left(g^{(j)}, h^{(j)}\right)'$ are:

$$\sigma^{2(j)}|\{\mu_t^{(j)}\}_{t=1}^T \sim IG\left(\frac{\nu_0 + T - 1}{2}, \frac{\nu_0\sigma_0^2 + \boldsymbol{\mu}_{2:T}'^{(j)}\boldsymbol{\mu}_{2:T}^{(j)} + \boldsymbol{\tau}_0'\boldsymbol{\Lambda}_0\boldsymbol{\tau}_0 - \boldsymbol{\tau}_n'\boldsymbol{\Lambda}_n\boldsymbol{\tau}_n}{2}\right),$$
$$\left(g^{(j)}, h^{(j)}\right)'|\{\mu_t^{(j)}\}_{t=1}^T \sim \mathcal{N}(\boldsymbol{\tau}_n, \sigma^{2(j)}\boldsymbol{\Lambda}_n^{-1}),$$

with $\boldsymbol{\Lambda}_n = \widetilde{\boldsymbol{\mu}}_{1:(T-1)}'^{(j)}\widetilde{\boldsymbol{\mu}}_{1:(T-1)}^{(j)} + \boldsymbol{\Lambda}_0$, and $\boldsymbol{\tau}_n = \boldsymbol{\Lambda}_n^{-1}\left(\widetilde{\boldsymbol{\mu}}_{1:(T-1)}'^{(j)}\boldsymbol{\mu}_{2:T}^{(j)} + \boldsymbol{\Lambda}_0\boldsymbol{\tau}_0\right)$.

For completeness, we also provide the update of latent state parameters when using the more parsimonious constraint, i.e. diagonal $\boldsymbol{X}'^{(j)}\boldsymbol{X}^{(j)}$. According to results from previous research (Krzanowski and Marriott, 1994b,a; Fokoué and Titterington, 2003), this constraint is one of the most parsimonious one, which only put constraints on $p(p-1)/2$ parameters. The constraint can also be implemented in MCMC by "unconstrained sampling-projection" procedure (Sen et al., 2018).

Without constraints, the sampling of $h^{(j)}$, $g^{(j)}$ and $\sigma^{2(j)}$ is the same as shown previously. The update of $\boldsymbol{A}^{(j)}$, $\boldsymbol{b}^{(j)}$ and $\boldsymbol{Q}^{(j)}$ is the standard multivariate Bayesian linear regression. Denote $\boldsymbol{X}_{2:T}^{(j)} = (\boldsymbol{x}_2^{(j)}, \ldots, \boldsymbol{x}_T^{(j)})'$ and $\widetilde{\boldsymbol{X}}_{1:(T-1)}^{(j)} = (\boldsymbol{1}_{T-1}, \boldsymbol{X}_{1:(T-1)}^{(j)})$, with $\boldsymbol{X}_{1:(T-1)}^{(j)} = (\boldsymbol{x}_1^{(j)}, \ldots, \boldsymbol{x}_{T-1}^{(j)})'$. Let us use the conjugate priors as following for $\boldsymbol{Q}^{(j)}$, $\boldsymbol{b}^{(j)}$ and $\boldsymbol{A}^{(j)}$:

$$\boldsymbol{Q}^{(j)} \sim \mathcal{W}^{-1}(\boldsymbol{\Psi}_0, \gamma_0),$$
$$vec((\boldsymbol{b}^{(j)}, \boldsymbol{A}'^{(j)})) \sim N(vec(\boldsymbol{T}_0), \boldsymbol{Q}^{(j)} \otimes \boldsymbol{\Gamma}_0^{-1}).$$

Here, the $\mathcal{W}^{-1}$ denotes the inverse-Wishart distribution. We can set the priors as $\boldsymbol{\Psi}_0 = 0.01\boldsymbol{I}_p$, $\gamma_0 = p+2$ and $\boldsymbol{T}_0 = (\boldsymbol{0}_p, \boldsymbol{I}_p)'$. Then, the full conditional distributions for $\boldsymbol{Q}^{(j)}$ and $vec((\boldsymbol{b}^{(j)}, \boldsymbol{A}^{(j)})')$ are:

$$\boldsymbol{Q}^{(j)}|\boldsymbol{X}^{(j)} \sim \mathcal{W}^{-1}(\boldsymbol{\Psi}_n, \gamma_n),$$
$$vec((\boldsymbol{b}^{(j)}, \boldsymbol{A}^{(j)})')|\boldsymbol{X}^{(j)} \sim N(vec(\boldsymbol{T}_n), \boldsymbol{Q}^{(j)} \otimes \boldsymbol{\Gamma}_n^{-1}),$$

with

$$
\begin{aligned}
\boldsymbol{\Psi}_n &= \boldsymbol{\Psi}_0 + (\boldsymbol{X}^{(j)}_{2:T} - \widetilde{\boldsymbol{X}}^{(j)}_{1:(T-1)}\boldsymbol{T}_n)'(\boldsymbol{X}^{(j)}_{2:T} - \widetilde{\boldsymbol{X}}^{(j)}_{1:(T-1)}\boldsymbol{T}_n), \\
&\quad + (\boldsymbol{T}_n - \boldsymbol{T}_0)'\boldsymbol{\Gamma}_0(\boldsymbol{T}_n - \boldsymbol{T}_0), \\
\gamma_n &= \gamma_0 + T - 1, \\
\boldsymbol{T}_n &= \boldsymbol{\Gamma}_n^{-1}(\widetilde{\boldsymbol{X}}'^{(j)}_{1:(T-1)}\boldsymbol{X}^{(j)}_{2:T} + \boldsymbol{\Gamma}_0\boldsymbol{T}_0), \\
\boldsymbol{\Gamma}_n &= \widetilde{\boldsymbol{X}}'^{(j)}_{1:(T-1)}\widetilde{\boldsymbol{X}}^{(j)}_{1:(T-1)} + \boldsymbol{\Gamma}_0.
\end{aligned}
$$

Before updating the cluster indices z_i, the sampling of $\{\boldsymbol{x}_t^{(j)}, \mu_t^{(j)}, \delta_i, \boldsymbol{c}_i, \boldsymbol{\theta}^{(j)}\}$ is repeated several times (5 times in both simulation and application for this paper). To save time, we can further allow the repetitions to be pre-stopped, when the training log-likelihood converges roughly.

Update z_i To update the cluster assignments for each neuron i, we use a partition based algorithm for MFM, similarly as described in Miller and Harrison 2018.

Let $\mathcal{C}$ denote a partition of neurons, and $\mathcal{C}\backslash i$ denote the partition obtained by removing neuron i from $\mathcal{C}$.

1. Initialize $\mathcal{C}$ and $\{\boldsymbol{\theta}^{(c)} : c \in \mathcal{C}\}$ (e.g. one cluster).
2. Repeat the following steps G times to obtain G samples. For $i = 1, \ldots, N$: remove neuron i from $\mathcal{C}$ and place it:
   (a) in $c \in \mathcal{C}\backslash i$ with probability $\propto (|c| + \gamma) M_{\boldsymbol{\theta}^{(c)}}(\boldsymbol{y}_i)$, where γ is defined in the MFM model in the main text (Equation 2) and $M_{\boldsymbol{\theta}^{(c)}}(\boldsymbol{y}_i)$ denotes the marginal likelihood of neuron i in cluster c, when integrating the loading $\boldsymbol{c}_i$ out (Equation (3)). The marginal likelihood is approximated by using Poisson-Gamma conjugacy.
   (b) in a new cluster c^* with probability $\propto \gamma \frac{V_n(t+1)}{V_n(t)} M_{\boldsymbol{\theta}^{(c^*)}}(\boldsymbol{y}_i)$, where t is the number of partitions obtained by removing the neuron i and $V_n(t) = \sum_{j=1}^{\infty} \frac{j_{(t)}}{(\gamma j)^{(n)}} f_k(j)$, with $x^{(m)} = x(x+1)\cdots(x+m-1)$, $x_{(m)} = x(x-1)\cdots(x-m+1)$, $x^{(0)} = 1$ and $x_{(0)} = 1$.

The update is an adaptation of partition-based algorithm for DPM (Neal, 2000), but with two substitutions: 1) replace $|c_i|$ by $|c_i| + \gamma$ and 2) replace α by $\gamma V_n(t+1)/V_n(t)$. See more details and discussions in (Miller and Harrison, 2018). When evaluating the likelihood, we marginalize the cluster-independent loading $\boldsymbol{c}_i$ out. This is necessary for the high dimensional situation, otherwise the chain will stop moving.

One issue with incremental Gibbs samplers such as Algorithm 3 and 8 in Neal (2000), when applied to DPM, is that mixing can be somewhat slow. To further improve the mixing, we may intersperse the "split-merge" Metropolis-Hasting updates (Jain and Neal, 2007, 2004) between Gibbs sweeps, as in (Miller and Harrison, 2018).

A.2 Sample Latent Vectors Using Pólya-Gamma Augmentation and Metropolis-Hastings Algorithm

In this section, we provide details of sampling algorithms to draw the latent states $\boldsymbol{X}^{(j)}$ and population baseline $\boldsymbol{\mu}^{(j)}$ from the exact posterior of the model. In order to explain the algorithm much clearer, we just focus on a given cluster index, and thus in this section we suspend the superscript (j) as the cluster index in our notations. We present the sampling idea with two major parts as described below.

Pólya-Gamma Augmentation Although the mixDPFA model does not directly follow the PG augmentation scheme (Polson et al., 2013), we can approximate the Poisson distribution by a negative binomial (NB) distribution, i.e., $\lim_{r\to\infty} \text{NB}(r, \sigma(\psi - \log r)) = \text{Poisson}(e^{\psi})$, where $\sigma(\psi) = e^{\psi}/(1+e^{\psi})$ and $\text{NB}(r,p)$ denotes the NB distribution with $rp/(1-p)$ as its expectation. We approximate the proposed DPFA model using a NB, then we can instead sample a mixture of NB factor analyzers using the PG scheme (Windle et al., 2013). Further, we use the forward-filtering-backward-sampling (FFBS) algorithm (Carter and Kohn, 1994; Frühwirth-Schnatter, 1994) to update

the latent states $\boldsymbol{X}$ and population baseline $\boldsymbol{\mu}$. These two steps are summarized in the step 1 and step 2 of Algorithm 1 for updating $\widetilde{\boldsymbol{X}} = (\boldsymbol{\mu}, \boldsymbol{X})$.

Metropolis-Hastings Step Next, we use the samples of $\widetilde{\boldsymbol{X}}$ yielded from FFBS algorithm as a proposal, where we employ a Metropolis-Hastings (MH) step to reject or accept the proposal. In this step, the dispersion parameter r in NB distribution becomes a tuning parameter, to balance acceptance rate and autocorrelation in MH. When r is large, the approximation to Poisson observation is accurate and the MH performs similar to the Gibbs sampler. Here, we allow neurons at different time points to have unique tuning parameters r_{it}. The MH step is summarized in step 3 of Algorithm 1.

We have further implemented Algorithm 1 to fit DPFA model for cluster 1 data in Fig. 2 and the results have been presented in Fig. 1. We set the tuning parameters for each neuron at all time points as $r_{it} = 10$, to tune the acceptance rate around 0.4. When implementing the PG-MH algorithm in DPFA, the ω_{it} (i.e., PG random variable) was sampled using `pgdraw` function in R package `pgdraw`, which is called from our MATLAB code. Although we can program Algorithm 1 without calling `pgdraw` function from R to save some time, to sample the PG random variable is often much computationally expensive in comparison to sample from the Gaussian random variable. When fitting the simulated data with $N = 5$, $T = 1000$ and $p = 2$ (cluster 1 data in Fig. 2) using a 3.40 GHz processor with 16 GB of RAM, it takes 131.64s for PG-MH to draw 1000 posterior samples, while the proposed approximation method in the main context of the paper takes 83.86s. Although the proposed method takes the approximation to the full conditional of the latent state $\boldsymbol{X}$ and population baseline $\boldsymbol{\mu}$, the results are similar as to sample directly from the exact full conditional. Table 1 shows the similarities, defined by cosine function, of fitted $\boldsymbol{\mu}$ and $\boldsymbol{X} = (\boldsymbol{X}_{*1}, \boldsymbol{X}_{*2})$ between the PG-MH and the Gaussian approximation for full conditional, where $\boldsymbol{X}_{*l}$ denotes the l-th latent vectors (i.e. the l-th column of $\boldsymbol{X}$). Here, $\hat{\boldsymbol{\xi}}_{\text{PGMH}}$ denotes the average from iteration 5000 to 10,000 for chain 1 of PG-MH (Fig. 1), $\hat{\boldsymbol{\xi}}_{\mathcal{N}}$ denotes the average from iteration 5000 to 10,000 for the chain in Fig. 2 of the Gaussian approximation, and $\boldsymbol{\xi}_{\text{true}}$ denotes the ground truth. In each column, $\boldsymbol{\xi}$ is replaced by $\boldsymbol{\mu}$, $\boldsymbol{X}_{*1}$ and $\boldsymbol{X}_{*2}$ respectively. The posterior means for these two method are close to each other, i.e. $\cos(\hat{\boldsymbol{\xi}}_{\text{PGMH}}, \hat{\boldsymbol{\xi}}_{\mathcal{N}}) \approx 1$.

A.3 Supplementary Results for Neuropixels Application

This section shows supplementary results when applying the proposed model to the Neuropixels dataset (4 Multi-region neural spike recordings). Here, we show 1) results sorted by maximum a posteriori probability (MAP) estimates and 2) clustering results in another independent chain (Fig. 2).

Algorithm 1: Pólya-Gamma-Metropolis-Hastings Algorithm (PG-MH) for Poisson Dynamic Model

Recall the DPFA model using the notations defined in A.1:

$$y_{it} \sim Poi(\lambda_{it}),$$
$$\log \lambda_{it} = \delta_i + \widetilde{\boldsymbol{c}}_i' \widetilde{\boldsymbol{x}}_t,$$

for $i = 1, \ldots, n$ and $t = 1, \ldots, T$. Here, $\widetilde{\boldsymbol{c}}_i = (1, \boldsymbol{c}_i')'$ and $\widetilde{\boldsymbol{x}}_t = (\mu_t, \boldsymbol{x}_t')'$. $\widetilde{\boldsymbol{x}}_t$ follows linear dynamics $\widetilde{\boldsymbol{x}}_{t+1} | \widetilde{\boldsymbol{x}}_t \sim \mathcal{N}(\widetilde{\boldsymbol{A}} \widetilde{\boldsymbol{x}}_t + \widetilde{\boldsymbol{b}}, \widetilde{\boldsymbol{Q}})$. Denote the prior as $\widetilde{\boldsymbol{x}}_1 \sim \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{V}_0)$. Given the sample from the $(G-1)$-th iteration $\widetilde{\boldsymbol{x}}_t^{(G-1)}$ and $\boldsymbol{U} = \{\widetilde{\boldsymbol{c}}_i, \delta_i, \widetilde{\boldsymbol{A}}, \widetilde{\boldsymbol{b}}, \widetilde{\boldsymbol{Q}}\}$.

1. sample ω_{it} from PG distribution and calculate $\hat{y}_{it}$, which follows $\mathcal{N}(\widetilde{\boldsymbol{c}}_i' \widetilde{\boldsymbol{x}}_t^{(G-1)}, \omega_{it}^{-1})$
for $t = 1, \ldots, T$ **do**
  for $i = 1, \ldots, n$ **do**
    sample $\omega_{it} \sim P_{PG}(r_{it} + y_{it}, \delta_i + \widetilde{\boldsymbol{c}}_i' \widetilde{\boldsymbol{x}}_t^{(G-1)} - \log r_{it})$
    $\kappa_{it} = (y_{it} - r_{it})/2 + \omega_{it}(\log r_{it} - \delta_i)$
    $\hat{y}_{it} = \omega_{it}^{-1} \kappa_{it}$
  end
end

2. Forward-filtering-backward-sampling (FFBS) for $\widetilde{\boldsymbol{X}}$
Denote $\hat{\boldsymbol{y}}_t = (\hat{y}_{1t}, \ldots, \hat{y}_{Nt})'$, $\boldsymbol{\Omega}_t = Diag([\omega_{1t}, \ldots, \omega_{Nt}])$ and $\widetilde{\boldsymbol{C}} = (\widetilde{\boldsymbol{c}}_1, \ldots, \widetilde{\boldsymbol{c}}_N)'$
for $t = 1, \ldots, T$ **do**
  $\boldsymbol{m}_{t|t-1} = \widetilde{\boldsymbol{A}} \boldsymbol{m}_{t-1} + \widetilde{\boldsymbol{b}}$
  $\boldsymbol{V}_{t|t-1} = \widetilde{\boldsymbol{A}} \boldsymbol{V}_{t-1} \widetilde{\boldsymbol{A}}' + \widetilde{\boldsymbol{Q}}$
  $\boldsymbol{K}_t = \boldsymbol{V}_{t|t-1} \widetilde{\boldsymbol{C}}' (\widetilde{\boldsymbol{C}} \boldsymbol{V}_{t|t-1} \widetilde{\boldsymbol{C}}' + \boldsymbol{\Omega}_t^{-1})^{-1}$
  $\boldsymbol{m}_t = \boldsymbol{m}_{t|t-1} + \boldsymbol{K}_t (\hat{\boldsymbol{y}}_t - \widetilde{\boldsymbol{C}} \boldsymbol{m}_{t|t-1})$
  $\boldsymbol{V}_t = (\boldsymbol{I} - \boldsymbol{K}_t \widetilde{\boldsymbol{C}}) \boldsymbol{V}_{t|t-1}$
end
sample $\widetilde{\boldsymbol{x}}_T^* \sim \mathcal{N}(\boldsymbol{m}_T, \boldsymbol{V}_T)$
for $t = T-1, \ldots, 1$ **do**
  $\boldsymbol{J}_t = \boldsymbol{V}_t \widetilde{\boldsymbol{A}}' (\widetilde{\boldsymbol{A}} \boldsymbol{V}_t \widetilde{\boldsymbol{A}}' + \widetilde{\boldsymbol{Q}})^{-1}$
  $\boldsymbol{m}_t^* = \boldsymbol{m}_t + \boldsymbol{J}_t (\boldsymbol{x}_{t+1}^* - \widetilde{\boldsymbol{A}} \boldsymbol{m}_t - \widetilde{\boldsymbol{b}})$
  $\boldsymbol{V}_t^* = (\boldsymbol{I} - \boldsymbol{J}_t \widetilde{\boldsymbol{A}}) \boldsymbol{V}_t$
  sample $\boldsymbol{x}_t^* \sim \mathcal{N}(\boldsymbol{m}_t^*, \boldsymbol{V}_t^*)$
end

3. Accept or reject the proposal $\widetilde{\boldsymbol{X}}^*$
compute the acceptance ratio

$$\begin{aligned}\zeta &= \frac{\pi(\widetilde{\boldsymbol{X}}^* | \{\boldsymbol{y}_i\}_{i=1}^n, \boldsymbol{U})}{\pi(\widetilde{\boldsymbol{X}}^{(G-1)} | \{\boldsymbol{y}_i\}_{i=1}^n, \boldsymbol{U})} \frac{q(\widetilde{\boldsymbol{X}}^{(G-1)} | \widetilde{\boldsymbol{X}}^*, \boldsymbol{U})}{q(\widetilde{\boldsymbol{X}}^* | \widetilde{\boldsymbol{X}}^{(G-1)}, \boldsymbol{U})} \\ &= \frac{P(\{\boldsymbol{y}_i\}_{i=1}^n | \widetilde{\boldsymbol{X}}^*)}{P(\{\boldsymbol{y}_i\}_{i=1}^n | \widetilde{\boldsymbol{X}}^{(G-1)})} \frac{NB(\{\boldsymbol{y}_i\}_{i=1}^n | \widetilde{\boldsymbol{X}}^{(G-1)}, \boldsymbol{R})}{NB(\{\boldsymbol{y}_i\}_{i=1}^n | \widetilde{\boldsymbol{X}}^*, \boldsymbol{R})},\end{aligned}$$

where $\boldsymbol{R} = \{r_{it}\}$ is matrix for dispersion parameters for each neuron at all time points. $P(\cdot)$ denotes the Poisson likelihood, and $NB(\cdot)$ denotes the negative binomial likelihood. Accept the proposal $\widetilde{\boldsymbol{X}}^*$ with probability $\min(1, \zeta)$.

Table 1: PG-MH vs. Gaussian approximation for $\boldsymbol{\mu}$ and $\boldsymbol{X}$

	$\boldsymbol{\mu}$	$\boldsymbol{X}_{*1}$	$\boldsymbol{X}_{*2}$
$\cos\left(\hat{\boldsymbol{\xi}}_{\text{PGMH}}, \boldsymbol{\xi}_{\text{true}}\right)$	0.9724	0.7663	0.9728
$\cos\left(\hat{\boldsymbol{\xi}}_{\mathcal{N}}, \boldsymbol{\xi}_{\text{true}}\right)$	0.9680	0.7443	0.9699
$\cos\left(\hat{\boldsymbol{\xi}}_{\text{PGMH}}, \hat{\boldsymbol{\xi}}_{\mathcal{N}}\right)$	0.9435	0.9664	0.9959

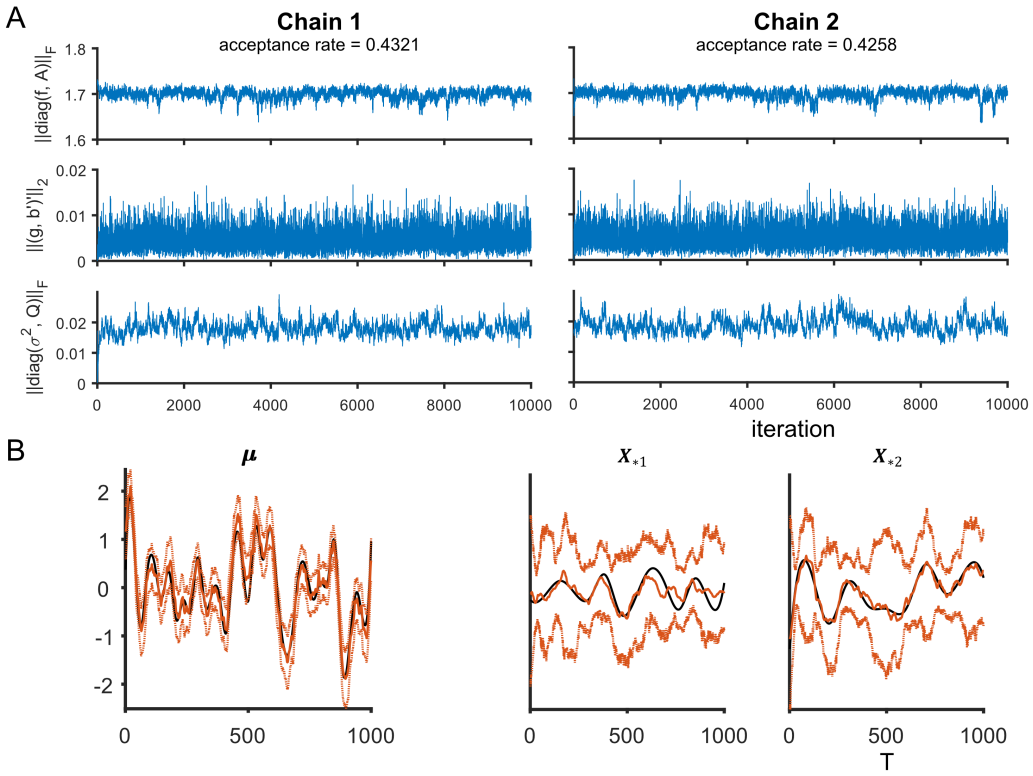

Figure 1: **PG-MH for DPFA** Two independent chains (10,000 iterations) for cluster 1 data in Figure 2. The full conditional distributions of $\boldsymbol{\mu}$ and $\boldsymbol{X}$ are sampled by PG-MH algorithm (Algorithm 1).**A.** Traceplot of Frobenius norms of linear dynamics. The acceptance rates in subtitle are for sampling $\boldsymbol{\mu}$ and $\boldsymbol{X}$ (PG-MH step). **B.** The true (black) and fitted (colored) population baseline and latent factor. Use samples from iteration 5000 to 10,000 in chain 1, the solid orange lines show averages and the dashed lines show 95% HPD intervals.

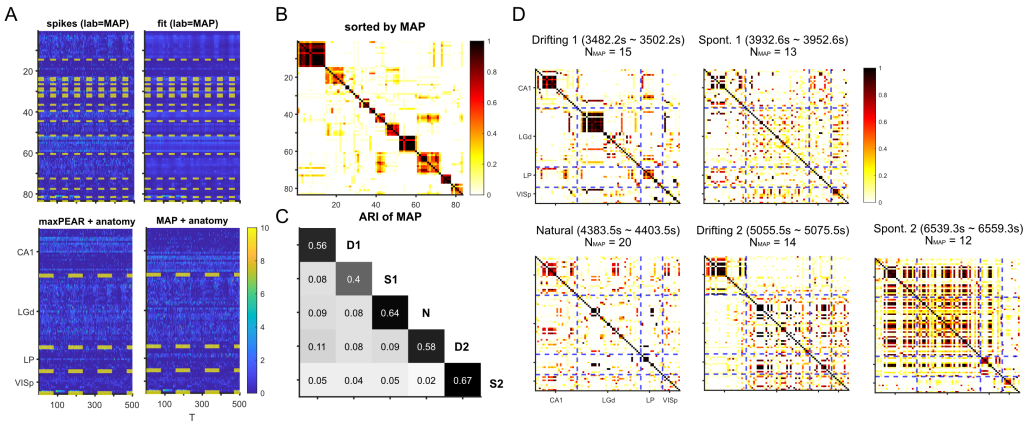

Figure 2: **Supplementary Results for Neuropixels Application A.** The first row shows the spike counts and fitted mean firing rate, sorted according to the MAP label estimates. The second row sort the spikes further according the anatomical sites, which shows clustering structure within each region. **B.** The posterior similarity matrix sorted by MAP estimates. **C.** The ARI of MAP estimates. The diagonal is ARI between 2 chains, and the off-diagonal is mean ARI of MAP for 4 combinations. **D.** Posterior similarity matrices for the second chains. Neurons are sorted as in the first panel of Fig. 4E

References

Carter, C. K. and Kohn, R. (1994). On Gibbs sampling for state space models. *Biometrika*, 81(3):541–553.

Duane, S., Kennedy, A., Pendleton, B. J., and Roweth, D. (1987). Hybrid monte carlo. *Physics Letters B*, 195(2):216–222.

Eden, U. T., Frank, L. M., Barbieri, R., Solo, V., and Brown, E. N. (2004). Dynamic Analysis of Neural Encoding by Point Process Adaptive Filtering. *Neural Computation*, 16(5):971–998.

Fokoué, E. and Titterington, D. M. (2003). Mixtures of factor analysers. Bayesian estimation and inference by stochastic simulation. *Machine Learning*, 50(1-2):73–94.

Frühwirth-Schnatter, S. (1994). Data augmentation and dynamic linear models. *Journal of Time Series Analysis*, 15:183–202.

Jain, S. and Neal, R. M. (2004). A split-merge markov chain monte carlo procedure for the dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13(1):158–182.

Jain, S. and Neal, R. M. (2007). Splitting and merging components of a nonconjugate Dirichlet process mixture model. *Bayesian Analysis*, 2(3):445–472.

Krzanowski, W. J. and Marriott, F. H. C. (1994a). Multivariate analysis. Part 2: Classification, covariance structures and repeated measurements.

Krzanowski, W. J. and Marriott, F. H. C. (1994b). Multivariate analysis. Part I. Distributions, ordination and inference.

Macke, J. H., Buesing, L., Cunningham, J. P., Yu, B. M., Shenoy, K. V., and Sahani, M. (2011). Empirical models of spiking in neural populations. *Advances in Neural Information Processing Systems*, 24.

Miller, J. W. and Harrison, M. T. (2018). Mixture models with a prior on the number of components. *Journal of the American Statistical Association*, 113(521):340.

Neal, R. M. (2000). Markov Chain Sampling Methods for Dirichlet Process Mixture Models. *Journal of Computational and Graphical Statistics*, 9(2):249–265.

Paninski, L., Ahmadian, Y., Ferreira, D. G., Koyama, S., Rahnama Rad, K., Vidne, M., Vogelstein, J., and Wu, W. (2010). A new look at state-space models for neural data. *Journal of Computational Neuroscience*, 29(1-2):107–126.

Polson, N. G., Scott, J. G., and Windle, J. (2013). Bayesian inference for logistic models using pólya–gamma latent variables. *Journal of the American Statistical Association*, 108(504):1339–1349.

Rauch, H. E., Tung, F., and Striebel, C. T. (1965). Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450.

Sen, D., Patra, S., and Dunson, D. (2018). Constrained inference through posterior projections.

Windle, J., Carvalho, C. M., Scott, J. G., and Sun, L. (2013). Efficient Data Augmentation in Dynamic Models for Binary and Count Data.