# Model-based Clustering for Neural Populations

The goal for this research is to do model-based clustering for neural populations, by making use of features for each counting process observation.

## 1 Notations

Assume we can observe neural activities for $N$ neurons, with counting observation up to $T$ steps. Therefore, the observation is a $N$-*by*-$T$ matrix, $\mathbf{Y} \in \mathbb{Z}_{\geq 0}^{N \times T}$ ,with each row represents the recording from single neuron. Denote the recording for neuron $i$ as $\mathbf{y}_i = (y_{i1}, \ldots, y_{\mathrm{iT}})'$, $i = 1, \ldots N.$, with the cluster index for neuron $i$ as $z_i \in \{1, \ldots\}$. The number of neurons in cluster $j$ is $n_j = \sum_{i=1}^{N} I(z_i = j)$, and $\sum_{j=1,2,\ldots} n_j = N$. The proportion/ probability in cluster $z_i$ is $\rho_{z_i}$.

## 2 Clustering Wrapper

The model-based clustering problem can be transformed into fitting the mixture model (MM). The likelihood for each cluster depends on how we model the counting observation, but fitting strategies for MM are the same for all models. Here, I choose to fit the MM by Gibbs sampler. Depending on whether the number of cluster is finite or not, there are two versions: finite mixture model (FMM) and Dirichlet process mixture model (DPMM).

### 2.1 Finite Mixture Model

Assume the number of cluster is $J$. The full likelihood for these $N$ neurons is

$$L = \prod_{i=1}^{N} \rho_{z_i} f\left(\mathbf{y}_i | \mathbf{\Theta}_{z_i}\right) = \prod_{j=1}^{J} \rho_j^{n_j} \left[ \prod_{i: z_i = j} f\left(\mathbf{y}_i | \mathbf{\Theta}_j\right) \right]$$

, where $\mathbf{\Theta}_j$ contains all parameters in cluster $j$ defined by the specific model. Therefore, the parameters need to update are:

(1) Cluster indicator: $\{z_i\}_{i=1}^{N}$

(2) Cluster proportion: $\rho = (\rho_1, \ldots \rho_J)'$

(3) Model parameters: $\mathbf{\Theta}_j$

The (conditional) priors for clustering-related parameters:

(1) Cluster indicator $\{z_i\}_{i=1}^N$: $P\left(z_i = j\right) = \rho_j$

(2) Cluster proportion $\rho = (\rho_1, \ldots \rho_J)'$:

$$\rho \sim Dir(\delta_1, \ldots \delta_J)$$

, where $\delta_1 = \ldots = \delta_J = 1$

So, the MCMC( Gibbs sampler) iteration for FMM is:

(1) Update $\{z_i\}_{i=1}^N$:

$$P\left(z_i = j \middle| \mathbf{y}_i, \{\mathbf{\Theta}_j\}_{j=1}^J\right) \propto \rho_j f\left(\mathbf{y}_i | \mathbf{\Theta}_j\right)$$

(2) Update $\rho = (\rho_1, \ldots \rho_J)^{'}$:

$$\rho | \ \{\mathbf{y}_i\}_{i=1}^N, \ \{z_i\}_{i=1}^N, \{\mathbf{\Theta}_j\}_{j=1}^J \sim Dir(\delta_1 + n_1, \ldots \delta_J + n_J)$$

(3) Update $\mathbf{\Theta}_j$: this is defined by the specific model. When there's no $z_i = j$, just sample $\mathbf{\Theta}_j$ from priors or by other observation-independent ways.

## 2.2 Dirichlet Process Mixture Model

Since calculation of posterior predictive distribution can be hard or even impossible for complicated models, instead of using the popular CRP representation of DP (Neal, 2020), I choose to use the slice sampler (Walker, 2007).

Use the "stick-breaking" construction for cluster proportion, i.e.

$$\rho_1 = \eta_1$$

$$\rho_j = (1 - \eta_1) \cdot \ldots \cdot (1 - \eta_{j-1}) \, \eta_j$$

$$\eta_j \sim Beta(1, \alpha)$$

In the slice sampler for DPMM, the parameters need to update are:

(1) "stick-breaking" elements: $\eta_j$

(2) Augment latent variable: $\{u_i\}_{i=1}^N$

(3) Model parameters: $\mathbf{\Theta}_j$

(4) Cluster indicator: $\{z_i\}_{i=1}^N$

So, the MCMC( Gibbs sampler) iteration for DPMM is:

(1) update $\eta_j$, for $j = 1, \ldots, z^* = max\left\{z_i\right\}_{i=1}^N$ as

$$\eta_j | \left\{z_i\right\}_{i=1}^N, \ldots \sim Beta(n_j + 1,\ N - \sum_{l=1}^{j} n_l + \alpha)$$

(2) update $\left\{u_i\right\}_{i=1}^N$:

$$u_i|\rho, \ldots \sim U(0,\ \rho_{z_i})$$

(3) update $\eta_j$, for $j = z^* + 1, \ldots,\ s^*$. $s^*$ is the smallest value, s.t. $\sum_{j=1}^{s^*} \rho_j > 1 - \min\left\{u_1, \ldots, u_N\right\}$

$$\eta_j \sim Beta(1, \alpha)$$

(4) Update $\mathbf{\Theta}_j$: this is defined by the specific model. When there's no $z_i = j$, just sample $\mathbf{\Theta}_j$ from priors or by other observation-independent ways.

(5) Update $\left\{z_i\right\}_{i=1}^N$

$$P\left(z_i = j \middle| \mathbf{y}_i, \left\{\mathbf{\Theta}_j\right\}, \rho, \left\{u_i\right\}_{i=1}^N\right) = \frac{f\left(\mathbf{y}_i|\mathbf{\Theta}_j\right)}{\sum_{j:\rho_j > u_i} f\left(\mathbf{y}_i|\mathbf{\Theta}_j\right)}$$

# 3 linear Dynamical System Model

Here, I model the observations by a linear dynamical system (LDS) model.

LDS models the multi-dimensional time series using a lower dimensional latent representation of the system, which evolves over time according to linear dynamics. By specifying the linear dynamics and process noise covariance, we can also handle the interactions between different neural populations (clusters).

## 3.1 Model Details

Denote the latent vector in cluster $j$ as $\mathbf{x}_t^{(j)} \in \mathbb{R}^{p_j}$. For simplicity, assume all $p_j = p$. Each observation follows a Poisson distribution:

$$\log \lambda_{\text{it}} = d_i + \mathbf{c'}_i \mathbf{x}_t^{(z_i)}$$

$$y_{\text{it}} \sim Poisson(\lambda_{\text{it}})$$

, where $\mathbf{c}_i \in \mathbb{R}^p$ and $\mathbf{x}_t^{(z_i)} \in \mathbb{R}^p$.

Although the loading ($d_i$ and $\mathbf{c}_i$) is determined by neuron index $i$, the distribution is also cluster-dependent. That is,

$$(d_i, \mathbf{c}_i')' \sim N(\boldsymbol{\mu}_{\text{dc}}^{(z_i)}, \mathbf{\Sigma}_{\text{dc}}^{(z_i)})$$

By doing this, the loading within each cluster is also correlated.

Denote all latent states as $\mathbf{x}_t = \left(\mathbf{x'}_t^{(1)}, \mathbf{x'}_t^{(2)}, \ldots\right)'$ and they evolve linearly with a Gaussian noise:

$$\mathbf{x}_1 \sim N(\mathbf{x}_0, \mathbf{Q}_0)$$
$$\mathbf{x}_{t+1}|\mathbf{x}_t \sim N(\mathbf{A}\mathbf{x}_t + \mathbf{b}, \mathbf{Q})$$

For simplicity, assume $\mathbf{Q}_0$ is known (e.g. $\mathbf{Q}_0 = \mathbf{I} \times 10^{-2}$).

If we assume process noise covariance is block diagonal (Joshua et al., 2020), we can write things as:

$$\mathbf{x}_{t+1}^{(j)}|\mathbf{x}_t^{(1)}, \mathbf{x}_t^{(2)}, \ldots \sim N(\sum_{l=1,\ldots} \mathbf{A}_{j\leftarrow l}\mathbf{x}_t^{(l)} + \mathbf{b}_j, \mathbf{Q}^{(j)})$$

Notice $\{\mathbf{A}_{j\leftarrow l}\}$ forms the full transition matrix as:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{1\leftarrow 1} & \mathbf{A}_{1\leftarrow 2} & \ldots \\ \mathbf{A}_{2\leftarrow 1} & \mathbf{A}_{2\leftarrow 2} & \ldots \\ \ldots & \ldots & \ldots \end{pmatrix}$$

Denote the $j^{\text{th}}$ row block of $\mathbf{A}$ as $\mathbf{A}_j = \begin{pmatrix} \mathbf{A}_{j\leftarrow 1} & \mathbf{A}_{j\leftarrow 2} & \ldots \end{pmatrix}$. Then, $\sum_{l=1,\ldots} \mathbf{A}_{j\leftarrow l}\mathbf{x}_t^{(l)} + \mathbf{b}_j = \mathbf{A}_j\mathbf{x}_t + \mathbf{b}_j$.

If we further let $\mathbf{Q}$ be diagonal, with the $k^{\text{th}}$ row of $\mathbf{x}_t$, $\mathbf{A}$, $\mathbf{b}$ denoted as $x_{\text{kt}}$, $\mathbf{a}_k$, $b_k$. The corresponding process noise variance is $q_k$. Then:

$$x_{k,t+1}|x_{\text{kt}} \sim N\left(\mathbf{a}_k^{'}\mathbf{x}_t + b_k, q_k\right)$$

To facilitate derivation in Gibbs sampler, write the linear dynamics of $\mathbf{x}_t$ in MV-GLM form, i.e.

$$\mathbf{x}_{t+1}^{'} = \begin{pmatrix} 1 & \mathbf{x}_t^{'} \end{pmatrix} \begin{pmatrix} \mathbf{b}^{'} \\ \mathbf{A}^{'} \end{pmatrix} + \epsilon_i^{'}$$

, where $\epsilon \sim N(0, \mathbf{Q})$. Then if we stack the latent by row, the problem is reduced to a multivariate general linear model (MV-GLM) problem:

$$\begin{pmatrix} \mathbf{x}_2^{'} \\ \mathbf{x}_3^{'} \\ \vdots \\ \mathbf{x}_T^{'} \end{pmatrix} = \begin{pmatrix} 1 & \mathbf{x}_1^{'} \\ 1 & \mathbf{x}_2^{'} \\ \vdots & \vdots \\ 1 & \mathbf{x}_{T-1}^{'} \end{pmatrix} \begin{pmatrix} \mathbf{b}^{'} \\ \mathbf{A}^{'} \end{pmatrix} + \mathbf{E}$$

, where $\mathbf{E} = (\epsilon_1, \ldots, \epsilon_{T-1})^{'}$.

Since the progress noise is independent at each step, $f(\mathbf{y}_i|\mathbf{\Theta}_j) = \prod_{t=1}^{T} P(y_{\text{it}}|\mathbf{\Theta}_j)$, where $P(\cdot)$ is the Poisson density and $\mathbf{\Theta}_j$ contains all parameters in cluster $j$.

## 3.2 Conditional Priors for Parameters

The parameters need to estimate:

(1) Latent vectors: $\{\mathbf{x}_t\}_{t=1}^T$

(2) Initials: $\mathbf{x}_0$

(3) Linear mapping (loading) for latent vectors: $\{d_i\}_{i=1}^N$ and $\{\mathbf{c}_i\}_{i=1}^N$

(4) Mean and covariance for loading in each cluster: $\left\{\boldsymbol{\mu}_{\text{dc}}^{(j)}\right\}_j$ and $\left\{\boldsymbol{\Sigma}_{\text{dc}}^{(j)}\right\}_j$

(5) Linear dynamics for latent vectors: $\mathbf{A}$ and $\mathbf{b}$

(6) Process noise: $\mathbf{Q}$

The conditional priors for these parameters:

(1) Latent vectors $\{\mathbf{x}_t\}_{t=1}^T$: the conditional prior is defined by

$$\mathbf{x}_1 \sim N(\mathbf{x}_0, \mathbf{Q}_0)$$

$$\mathbf{x}_{t+1}|\mathbf{x}_t \sim N(\mathbf{A}\mathbf{x}_t + \mathbf{b}, \mathbf{Q})$$

(2) Initials $\mathbf{x}_0$: assume there are $J$ clusters,

$$\mathbf{x}_0 \sim N(\boldsymbol{\mu}_{\mathbf{x}_{00}},\ \boldsymbol{\Sigma}_{\mathbf{x}_{00}})$$

, where $\boldsymbol{\mu}_{\mathbf{x}_{00}} = \mathbf{0}_{\text{Jp}}$ and $\boldsymbol{\Sigma}_{\mathbf{x}_{00}} = \mathbf{I}_{\text{Jp}}$

(3) Linear mapping (loading) for latent vectors $\{d_i\}_{i=1}^N$ and $\{\mathbf{c}_i\}_{i=1}^N$:

$$(d_i, \mathbf{c}_i')' \sim N(\boldsymbol{\mu}_{\text{dc}}^{(z_i)}, \boldsymbol{\Sigma}_{\text{dc}}^{(z_i)})$$

(4) Mean and covariance for loading in each cluster Mean and covariance for loading in each cluster $\left\{\boldsymbol{\mu}_{\text{dc}}^{(j)}\right\}_j$ and $\left\{\boldsymbol{\Sigma}_{\text{dc}}^{(j)}\right\}_j$:

$$\boldsymbol{\mu}_{\text{dc}}^{(j)} \sim N(\delta_{dc0}, \mathbf{T}_{dc0})$$

, where $\boldsymbol{\delta}_{dc0} = \mathbf{0}_{p+1}$ and $\mathbf{T}_{dc0} = \mathbf{I}_{p+1}$

$$\boldsymbol{\Sigma}_{\text{dc}}^{(j)} \sim W^{-1}\left(\Psi_{dc0}, \nu_{dc0}\right)$$

, where $\nu_{dc0} = p + 1 + 2$ and $\Psi_{dc0} = \mathbf{I}_{p+1} \times 10^{-4}$

(5) Linear dynamics for latent vectors: $\mathbf{A}$, $\mathbf{b}$ and process noise $\mathbf{Q}$.
Denote $\mathbf{F} = \begin{pmatrix} \mathbf{b}' \\ \mathbf{A}' \end{pmatrix}$, $\mathbf{f} = vec(\mathbf{F})$

$$
\begin{aligned}
P\left(\mathbf{f},\ \mathbf{Q}\right) = P(\mathbf{Q})P(\mathbf{f}|\mathbf{Q})\\
\mathbf{Q} \sim W^{-1}(\Psi_{\mathbf{Q}_0}, \nu_{\mathbf{Q}_0})\\
\mathbf{f} \sim N(\mathbf{f}_0, \mathbf{Q} \bigotimes \boldsymbol{\Lambda}_0^{-1})
\end{aligned}
$$

, where $\mathbf{f}_0 = vec(\mathbf{F}_0)$.If the number of cluster is $J$, $\nu_{\mathbf{Q}_0} = Jp + 2$, $\Psi_{\mathbf{Q}_0} = \mathbf{I}_{\mathrm{Jp}} \times 10^{-4}$. (To make the mean of $\mathbf{Q}$ loosely centered around $\mathbf{I}_{Jp} \times 10^{-4}$), $\mathbf{F}_0 = \begin{pmatrix} \mathbf{0'}_{Jp} \\ \mathbf{I}_{\mathrm{Jp}} \end{pmatrix}$ and $\boldsymbol{\Lambda}_0 = \mathbf{I}_{Jp+1}$

## 3.3 MCMC (Gibbs Sampler)

### 3.3.1 Update $\{\mathbf{x}_t\}_{t=1}^{T}$

Use Laplace approximation and make use of the block tri-diagonal Hessian.

Denote $t^{\mathrm{th}}$ column of mean firing rate and observation as $\widetilde{\boldsymbol{\lambda}}_t = (\lambda_{1t}, \ldots, \lambda_{\mathrm{Nt}})'$ and $\widetilde{\mathbf{y}}_t = (y_{1t}, \ldots, y_{\mathrm{Nt}})'$. The linear mapping matrix for all observations is $\mathbf{C}$, such that $\log \widetilde{\boldsymbol{\lambda}}_t = \mathbf{d}{+}\mathbf{C}\mathbf{x}_t$. Let $\mathbf{x} = (\mathbf{x'}_1, \ldots, \mathbf{x'}_T)'$and $f\left(\mathbf{x}\right) = \log P(\mathbf{x}|\left\{\mathbf{y}_i\right\}_{i=1}^{N}, \mathbf{C}, \mathbf{Q}_0, \mathbf{A}, \mathbf{b}, \mathbf{Q}, \ldots)$ The first and second derivative with respect to $\mathbf{x}$, for $t = 2, \ldots, T-1$:

$$
\begin{aligned}
\frac{\partial f}{\partial \mathbf{x}_1} &= \mathbf{C}'\left(\widetilde{\mathbf{y}}_1 - \widetilde{\boldsymbol{\lambda}}_1\right) - \mathbf{Q}_0^{-1}\left(\mathbf{x}_1 - \mathbf{x}_0\right) + \mathbf{A}'\mathbf{Q}^{-1}(\mathbf{x}_2 - \mathbf{A}\mathbf{x}_1 - \mathbf{b})\\
\frac{\partial f}{\partial \mathbf{x}_t} &= \mathbf{C}'\left(\widetilde{\mathbf{y}}_t - \widetilde{\boldsymbol{\lambda}}_t\right) - \mathbf{Q}^{-1}\left(\mathbf{x}_t - \mathbf{A}\mathbf{x}_{t-1} - \mathbf{b}\right) + \mathbf{A}'\mathbf{Q}^{-1}(\mathbf{x}_{t+1} - \mathbf{A}\mathbf{x}_t - \mathbf{b})\\
\frac{\partial f}{\partial \mathbf{x}_T} &= \mathbf{C}'\left(\widetilde{\mathbf{y}}_T - \widetilde{\boldsymbol{\lambda}}_T\right) - \mathbf{Q}^{-1}\left(\mathbf{x}_T - \mathbf{A}\mathbf{x}_{T-1} - \mathbf{b}\right)\\
\frac{\partial^2 f}{\partial \mathbf{x}_1 \partial \mathbf{x}_1'} &= -\mathbf{C}'\mathrm{Diag}\left(\widetilde{\boldsymbol{\lambda}}_1\right)\mathbf{C} - \mathbf{Q}_0^{-1} - \mathbf{A}'\mathbf{Q}^{-1}\mathbf{A}\\
\frac{\partial^2 f}{\partial \mathbf{x}_t \partial \mathbf{x}_t'} &= -\mathbf{C}'\mathrm{Diag}\left(\widetilde{\boldsymbol{\lambda}}_t\right)\mathbf{C} - \mathbf{Q}^{-1} - \mathbf{A}'\mathbf{Q}^{-1}\mathbf{A}\\
\frac{\partial^2 f}{\partial \mathbf{x}_T \partial \mathbf{x}_T'} &= -\mathbf{C}'\mathrm{Diag}\left(\widetilde{\boldsymbol{\lambda}}_T\right)\mathbf{C} - \mathbf{Q}^{-1}\\
\frac{\partial^2 f}{\partial \mathbf{x}_1 \partial \mathbf{x}_2'} &= \frac{\partial^2 f}{\partial \mathbf{x}_t \partial \mathbf{x}_{t+1}'} = \mathbf{A}'\mathbf{Q}^{-1} \qquad \frac{\partial^2 f}{\partial \mathbf{x}_t \partial \mathbf{x}_{t-1}'} = \mathbf{Q}^{-1}\mathbf{A}
\end{aligned}
$$

So, the gradient is:

$$
\nabla = \frac{\partial f}{\partial \mathbf{x}} = \left(\left(\frac{\partial f}{\partial \mathbf{x}_1}\right)', \ \ldots, \left(\frac{\partial f}{\partial \mathbf{x}_T}\right)'\right)'
$$

And the block tri-diagonal Hessian:

$$H = \frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{x}'} = \begin{pmatrix} \frac{\partial^2 f}{\partial \mathbf{x}_1 \partial \mathbf{x}_1'} & \mathbf{A}'\mathbf{Q}^{-1} & 0 & \cdots & 0 \\ \mathbf{Q}^{-1}\mathbf{A} & \frac{\partial^2 f}{\partial \mathbf{x}_2 \partial \mathbf{x}_2'} & \mathbf{A}'\mathbf{Q}^{-1} & \cdots & \vdots \\ 0 & \mathbf{Q}^{-1}\mathbf{A} & \frac{\partial^2 f}{\partial \mathbf{x}_3 \partial \mathbf{x}_3'} & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & \frac{\partial^2 f}{\partial \mathbf{x}_T \partial \mathbf{x}_T'} \end{pmatrix}$$

Use Newton-Raphson to find $\boldsymbol{\mu}_{\mathbf{x}} = \text{argmax}_{\mathbf{x}} \left( f \left( \mathbf{x} \right) \right)$ and $\boldsymbol{\Sigma}_{\mathbf{x}} = - \left[ \frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{x}'} \quad |_{\mathbf{X} = \boldsymbol{\mu}_{\mathbf{X}}} \right]^{\mathbf{-1}}$, such that $\left( P(\mathbf{x} | \left\{ \mathbf{y}_i \right\}_{i=1}^{N}, \ldots \right) \quad \approx N \left( \boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}} \right)$. When using Newton-Raphson (NR), $H \backslash \nabla$ in MATLAB will make use of block tri-diagonal structure automatically.

However, NR is not robust to bad initials. At the first few iterations, simply using fitting from previous step may lead to infinite Hessian. When the initial from previous step fails, use the approximation at recursive priors, , i.e. the adaptive smoother estimates, as the initial. The adaptive smoother estimates are from backward RTS smoother from adaptive filter, and the details about Poisson adaptive filter can be found in Eden et al., 2004.

To sample efficiently and make best use of sparse covariance, use Cholesky decomposition of $\boldsymbol{\Sigma}_{\mathbf{x}}^{-1} = \mathbf{R}\mathbf{R}'$: sample $\mathbf{Z} \sim N(\mathbf{R}'\mu_{\mathbf{x}}, \mathbf{I})$, then $\mathbf{x} = \left( \mathbf{R}' \right)^{-1} \mathbf{Z} \sim N(\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}})$.

### 3.3.2 Update $\mathbf{x}_0$

$$P \left( \mathbf{x}_0 | \mathbf{x}_1, \ \mathbf{Q}_0 \ldots \right) \propto N(\mathbf{x}_1 | \mathbf{x}_0, \ \mathbf{Q}_0) N(\mathbf{x}_0 | \boldsymbol{\mu}_{\mathbf{x}_{00}}, \ \boldsymbol{\Sigma}_{\mathbf{x}_{00}})$$

By conjugacy, $\mathbf{x}_0 | \mathbf{x}_1, \ \mathbf{Q}_0 \ldots \sim N(\boldsymbol{\mu}_{\mathbf{x}_0}, \ \boldsymbol{\Sigma}_{\mathbf{x}_0})$

$$\begin{aligned} \boldsymbol{\Sigma}_{\mathbf{x}_0} &= \left[ \boldsymbol{\Sigma}_{\mathbf{x}_{00}}^{-1} + \mathbf{Q}_0^{-1} \right]^{-1} \\ \boldsymbol{\mu}_{\mathbf{x}_0} &= \boldsymbol{\Sigma}_{\mathbf{x}_0} \left( \boldsymbol{\Sigma}_{\mathbf{x}_{00}}^{-1} \boldsymbol{\mu}_{\mathbf{x}_{00}} + \mathbf{Q}_0^{-1} \mathbf{x}_1 \right) \end{aligned}$$

### 3.3.3 Update $\{d_i\}_{i=1}^{N}$ and $\{\mathbf{c}_i\}_{i=1}^{N}$

To update efficiently, use Laplace approximation and the error is independent for each row (i.e. update things row by row). Denote $\left( d_i, \mathbf{c}_i' \right)' = \boldsymbol{\zeta}_i \in \mathbb{R}^{p+1}$ and

$\left(1, \mathbf{x}_t^{\prime(z_i)}\right) = \widetilde{\mathbf{x}}_t^{\prime(z_i)}$.

$$P\left(\boldsymbol{\zeta}_i \middle| \mathbf{y}_i,\ \left\{\mathbf{x}_t^{(z_i)}\right\}_{t=1}^{T}, \ldots\right) = \exp f\left(\boldsymbol{\zeta}_i\right) \approx N\left(\boldsymbol{\zeta}_i | \boldsymbol{\mu}_{\boldsymbol{\zeta}_i}, \boldsymbol{\Sigma}_{\boldsymbol{\zeta}_i}\ \right)$$

$$\frac{\partial f}{\partial \boldsymbol{\zeta}_i} = \frac{\partial l}{\partial \boldsymbol{\zeta}_i} - {\boldsymbol{\Sigma}_{\text{dc}}^{(z_i)}}^{-1}\left(\boldsymbol{\zeta}_i - \boldsymbol{\mu}_{\text{dc}}^{(z_i)}\right) = \left[\sum_{t=1}^{T} \widetilde{\mathbf{x}}_t^{(z_i)}\left(y_{\text{it}} - \lambda_{\text{it}}\right)\right] - {\boldsymbol{\Sigma}_{\text{dc}}^{(z_i)}}^{-1}\left(\boldsymbol{\zeta}_i - \boldsymbol{\mu}_{\text{dc}}^{(z_i)}\right)$$

$$\frac{\partial^2 f}{\partial \boldsymbol{\zeta}_i \partial \boldsymbol{\zeta}_i'} = \frac{\partial^2 l}{\partial \boldsymbol{\zeta}_i \partial \boldsymbol{\zeta}_i'} - {\boldsymbol{\Sigma}_{\text{dc}}^{(z_i)}}^{-1} = -\left[\sum_{t=1}^{T} \lambda_{\text{it}} \widetilde{\mathbf{x}}_t^{(z_i)} \widetilde{\mathbf{x}}_t^{\prime(z_i)}\right] - {\boldsymbol{\Sigma}_{\text{dc}}^{(z_i)}}^{-1}$$

, where $l$ is Poisson log-likelihood. Then use Newton-Raphson to find $\boldsymbol{\mu}_{\boldsymbol{\zeta}_i} = \text{argmax}_{\boldsymbol{\zeta}_i}\left(f\left(\boldsymbol{\zeta}_i\right)\right)$ and $\boldsymbol{\Sigma}_{\boldsymbol{\zeta}_i} = -\left[\frac{\partial^2 f}{\partial \boldsymbol{\zeta}_i \partial \boldsymbol{\zeta}_i'}\ |_{\boldsymbol{\zeta}_i = \boldsymbol{\mu}_{\boldsymbol{\zeta}_i}}\right]^{-1}$. If initial as the previous step fits fails, simply use prior mean of $\boldsymbol{\mu}_{\boldsymbol{\zeta}_i}$, i.e. $\boldsymbol{\delta}_{dc0}$.

DETOUR: we can write things in the form of MV-(Poisson)-GLM as follows (in cluster j with $n_j$ observations/ neurons, with corresponding loading be $\mathbf{d}_j$ and $\mathbf{C}_j$):

$$\log\begin{pmatrix} \lambda_{11} & \cdots & \lambda_{1n_j} \\ \vdots & \ddots & \vdots \\ \lambda_{T1} & \cdots & \lambda_{Tn_j} \end{pmatrix} = \begin{pmatrix} 1 & \mathbf{x}_1^{'} \\ \vdots & \vdots \\ 1 & \mathbf{x}_T^{'} \end{pmatrix}\begin{pmatrix} \mathbf{d}_j^{'} \\ \mathbf{C}_j^{'} \end{pmatrix} + \mathbf{E}_j$$

We can further organize things column-wise. let $\begin{pmatrix} 1 & \mathbf{x}_1^{'} \\ \vdots & \vdots \\ 1 & \mathbf{x}_T^{'} \end{pmatrix} = \mathbf{G}$, then the design matrix and corresponding log-mean response are $\begin{pmatrix} \mathbf{G} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{G} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \cdots & \mathbf{G} \end{pmatrix}$ and $\log\begin{pmatrix} \lambda_{11}^{(j)} & \cdots & \lambda_{T1}^{(j)} & \lambda_{12}^{(j)} & \cdots & \lambda_{T2}^{(j)} & \cdots & \lambda_{Tn_j}^{(j)} \end{pmatrix}^{'}$.

However, when I use the Laplace approximation and NR, because of the block-diagonal structure of design matrix, the error is always independent observation by observation. This is equivalent to observation-wise update as above.

**Maybe I did something wrong?** The point to say this is to make loading within each cluster depends on each other. Check these two later(ref1 and ref2).

Now I do it another way: make the prior cluster-dependent...

### 3.3.4 Update $\left\{\mu_{\mathbf{dc}}^{(j)}\right\}_j$ and $\left\{\boldsymbol{\Sigma}_{\mathbf{dc}}^{(j)}\right\}_j$

Purpose: to make loading within each cluster depends on each other, and this will help with clustering. As above, denote $\left(d_i, \mathbf{c}_i'\right)' = \boldsymbol{\zeta}_i \in \mathbb{R}^{p+1}$.

(1) Mean $\left\{\boldsymbol{\mu}_{\mathrm{dc}}^{(j)}\right\}_j$: by conjugacy, $\boldsymbol{\mu}_{\mathrm{dc}}^{(j)} \sim N\left(\boldsymbol{\delta}_{\mathrm{dc}}, \mathbf{T}_{\mathrm{dc}}\right)$

$$\mathbf{T}_{\mathrm{dc}}^{-1} = \left(\mathbf{T}_{dc0}^{-1} + n_j \boldsymbol{\Sigma}_{\mathrm{dc}}^{(j)^{-1}}\right)^{-1}$$
$$\boldsymbol{\delta}_{\mathrm{dc}} = \mathbf{T}_{\mathrm{dc}} \left(\mathbf{T}_{dc0}^{-1}\boldsymbol{\delta}_{dc0} + \boldsymbol{\Sigma}_{\mathrm{dc}}^{(j)^{-1}} \sum_{i:z_i=j} \boldsymbol{\zeta}_i\right)$$

(2) Covariance $\left\{\boldsymbol{\Sigma}_{\mathrm{dc}}^{(j)}\right\}_j$: by conjugacy, $\boldsymbol{\Sigma}_{\mathrm{dc}}^{(j)} \sim W^{-1}\left(\Psi_{\mathrm{dc}}, \nu_{\mathrm{dc}}\right)$

$$\nu_{\mathrm{dc}} = n_j + \nu_{dc0}$$
$$\Psi_{\mathrm{dc}} = \Psi_{dc0} + \sum_{i:z_i=j} \left(\boldsymbol{\zeta}_i - \boldsymbol{\mu}_{\mathrm{dc}}^{(j)}\right)\left(\boldsymbol{\zeta}_i - \boldsymbol{\mu}_{\mathrm{dc}}^{(j)}\right)'$$

### 3.3.5 Update A, b and Q

Here, I only give the update for full $\mathbf{Q}$. If we want to assume block-diagonal or diagonal structure of $\mathbf{Q}$, just update things cluster-by-cluster or latent-by-latent.

As shown before, the problem is the usual Bayesian MV-GLM problem. Denote $\mathbf{F} = \begin{pmatrix} \mathbf{b}' \\ \mathbf{A}' \end{pmatrix}$, $\mathbf{f} = vec(\mathbf{F})$, $\mathbf{Y_{bA}} = (\mathbf{x}_2, \mathbf{x}_3 \ldots, \mathbf{x}_T)'$ and $\mathbf{X_{bA}} = \begin{pmatrix} 1 & \mathbf{x}_1' \\ 1 & \mathbf{x}_2' \\ \vdots & \vdots \\ 1 & \mathbf{x}_{T-1}' \end{pmatrix}$

$$\mathbf{Y_{bA}} = \mathbf{X_{bA}}\mathbf{F} + \mathbf{E}$$

, where $\mathbf{E} = (\epsilon_1, \ldots, \epsilon_{T-1})'$ and $\epsilon \sim N(0, \mathbf{Q})$. Then posteriors are

$$\mathbf{Q}|\mathbf{Y}_{\mathrm{bA}}, \mathbf{X}_{\mathrm{bA}}, \ldots \sim W^{-1}(\Psi_{\mathbf{Q}}, \nu_{\mathbf{Q}})$$
$$\mathbf{f}|\mathbf{Y}_{\mathrm{bA}}, \mathbf{X}_{\mathrm{bA}}, \mathbf{Q}, \ldots \sim N(vec(\mathbf{F_{bA}}), \mathbf{Q} \bigotimes \boldsymbol{\Lambda}_{\mathbf{bA}}^{-1})$$

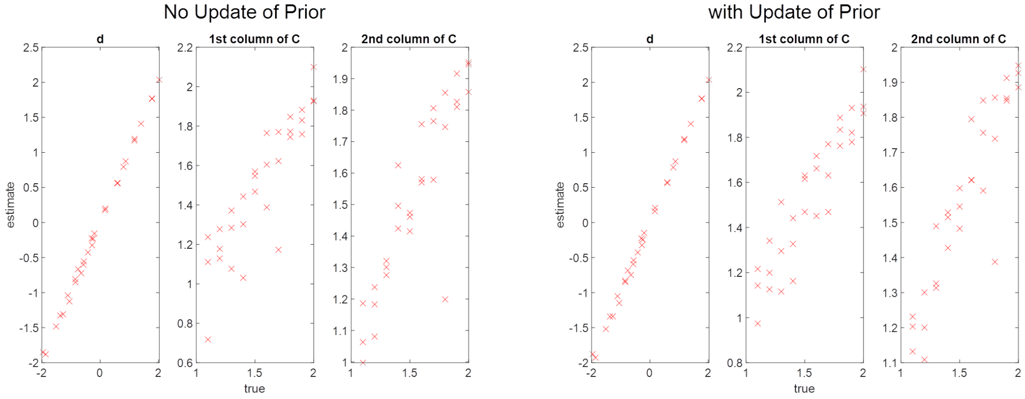
, with parameters be:

$$\Psi_{\mathbf{Q}} = \Psi_{\mathbf{Q}_0} + (\mathbf{Y_{bA}} - \mathbf{X_{bA}}\mathbf{F_{bA}})'(\mathbf{Y_{bA}} - \mathbf{X_{bA}}\mathbf{F_{bA}}) + (\mathbf{F_{bA}} - \mathbf{F}_0)'\boldsymbol{\Lambda}_0(\mathbf{F_{bA}} - \mathbf{F}_0)$$
$$\nu_{\mathbf{Q}} = \nu_{\mathbf{Q}_0} + T - 1$$
$$\mathbf{F_{bA}} = (\mathbf{X}_{\mathbf{bA}}'\mathbf{X_{bA}} + \boldsymbol{\Lambda}_0)^{-1}(\mathbf{X}_{\mathbf{bA}}'\mathbf{Y_{bA}} + \boldsymbol{\Lambda}_0\mathbf{F}_0)$$
$$\boldsymbol{\Lambda}_{\mathbf{bA}} = \mathbf{X}_{\mathbf{bA}}'\mathbf{X_{bA}} + \boldsymbol{\Lambda}_0$$

# 4 Model Check [Problems for b and A partially solved]

## 4.1 Estimation of d and C alone

First, turn off all others except for loading (related) parameters and set them to be true values. There are 2 versions: (1) update priors as in subsection 3.3.4 and (2) no update of priors, and just set the prior for each as the standard normal. For the estimation with prior update, the results are mean from iteration 50 to 100 (Figure 1).



(a)

(b)

Figure 1: estimation of loading

The mixing of convergence for loading priors (in subsection 3.3.4) are fine. The update of priors don't influence results a lot, but this will help clustering a lot. It seems estimation of loading is fine. Let's see what happens when estimation of latents $\mathbf{x}_t$ is on at the same time.

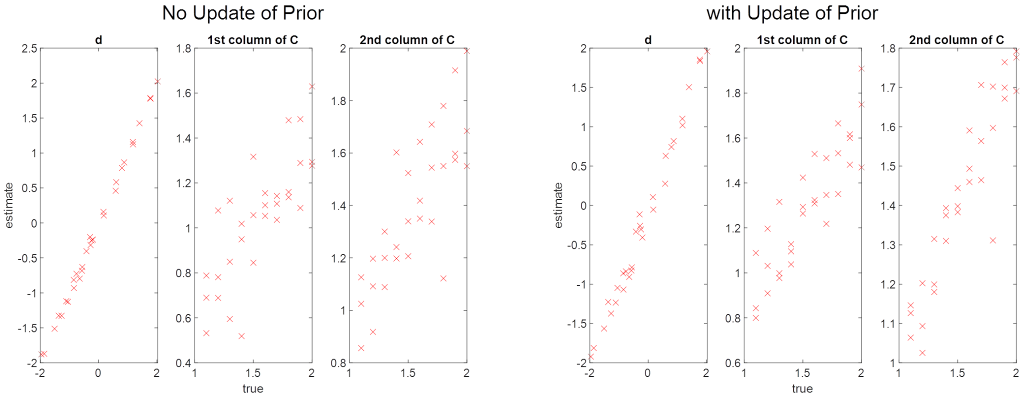
## 4.2 Estimation of d, C and $\mathbf{x}_t$

When the estimation of $\mathbf{x}_t$ is on, we need more iterations. The results are mean from iteration 500 to 1000 (Figure 2).

The estimation of loading:

And the estimation of corresponding latents (Figure 3):

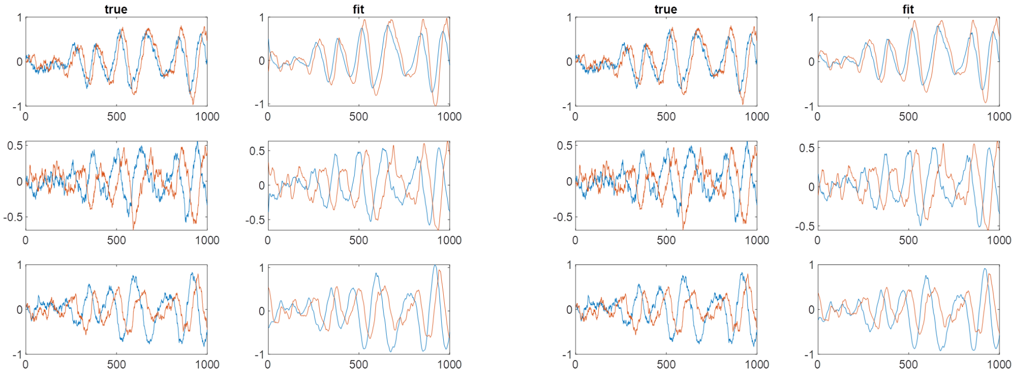
Things are still fine. The remaining parts may influence latent estimations are dynamics ($\mathbf{b}$, $\mathbf{A}$) and prior process noise $\mathbf{Q}$.



(a)

(b)

Figure 2: estimation of loading, with latents on



(a) no update of prior

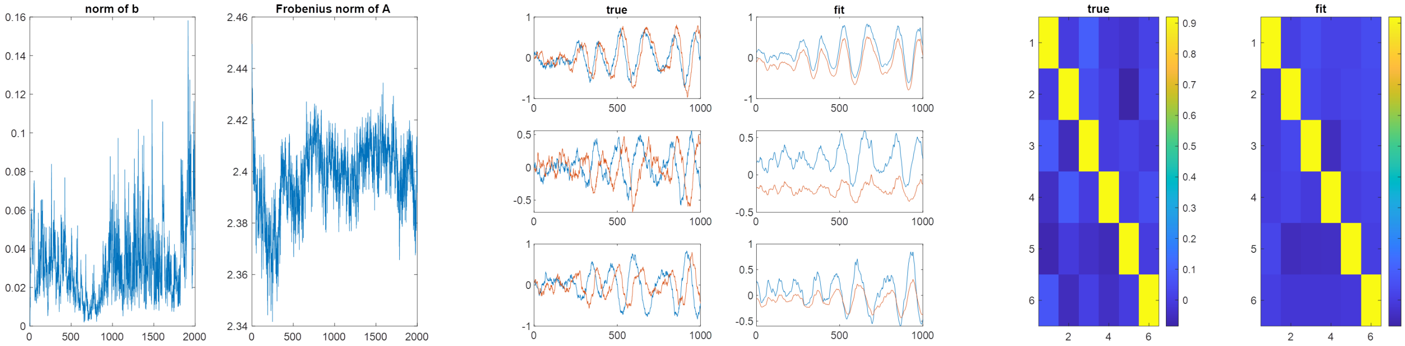
(b) with update of prior

Figure 3: estimation of loading

## 4.3 Estimation of b, A, Q and $\mathbf{x}_t$

I show the results of block-diagonal (Figure 4) and full (Figure 5) **Q**. These results are from 2000 MCMC samples. The latent and dynamics are from average from 500 to 2000 iterations.

Both block-diagonal and full looks fine, but the mixing of full process noise version looks better.

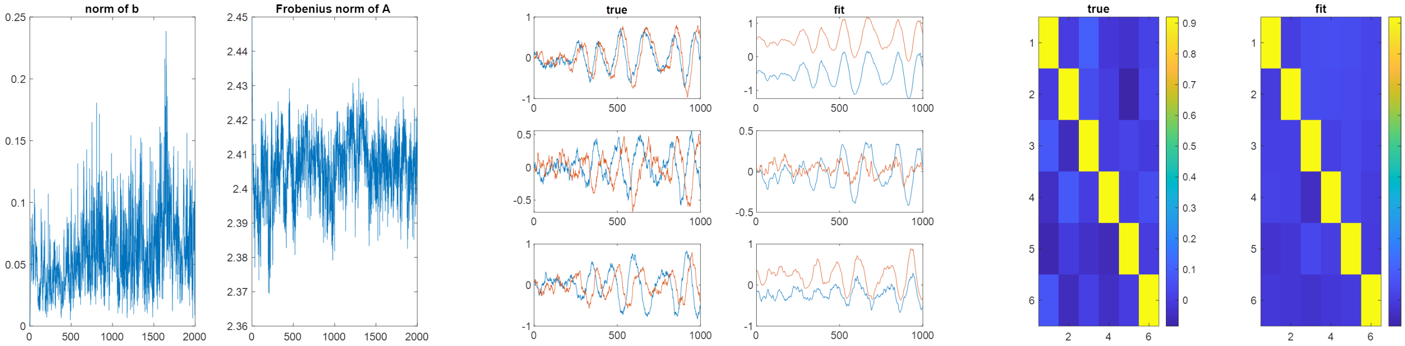


(a) trace

(b) latent

(c) dynamics **A**

Figure 4: sampling of $\mathbf{b}$, $\mathbf{A}$, $\mathbf{Q}$ and $\mathbf{x}_t$, block-diagonal process noise



(a) trace

(b) latent

(c) dynamics **A**

Figure 5: sampling of $\mathbf{b}$, $\mathbf{A}$, $\mathbf{Q}$ and $\mathbf{x}_t$, full process noise

# 5 Simulations

There are two set of simulation examples. The first example is generated from LDS model directly, while in the second example the latents are generated directly without explicit specifying linear dynamics. In all following results, I fit things with both block-diagonal and full $\mathbf{Q}$. The fitting results for block-diagonal version is a bit better, because the underlying true $\mathbf{Q}$ is diagonal. In the following part, I only show results from block-diagonal fitting for labeled data. For unlabeled data, i.e. clustering, all the results can be found in this folder.

## 5.1 Simulation 1: Generate from LDS Directly

In this simulation, there are 3 clusters with 10 neurons in each cluster. The dimension of latents in each cluster is 2. In the linear dynamics, the bias term **b** is zero. There's no within-population interaction but has some weak between-population interactions. In other words, the linear dynamics matrix **A** is roughly diagonal. The details of simulation can be found in the first section of the simulation 1.

### 5.1.1 Labeled Data: No Clustering

The code for fitting is here. After running 2000 iterations, the trace of the sample is shown in Figure 6.

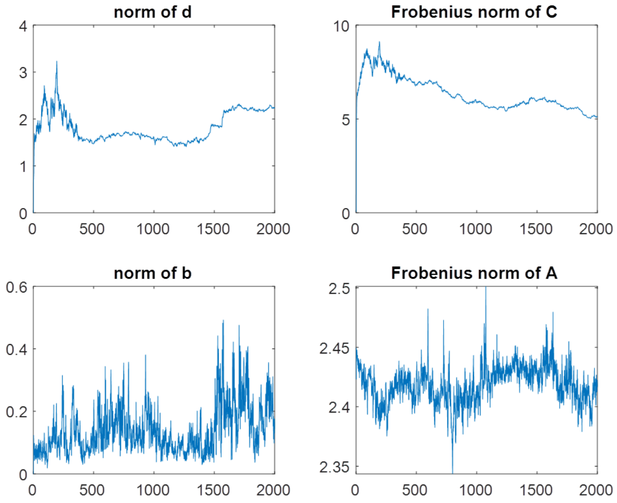


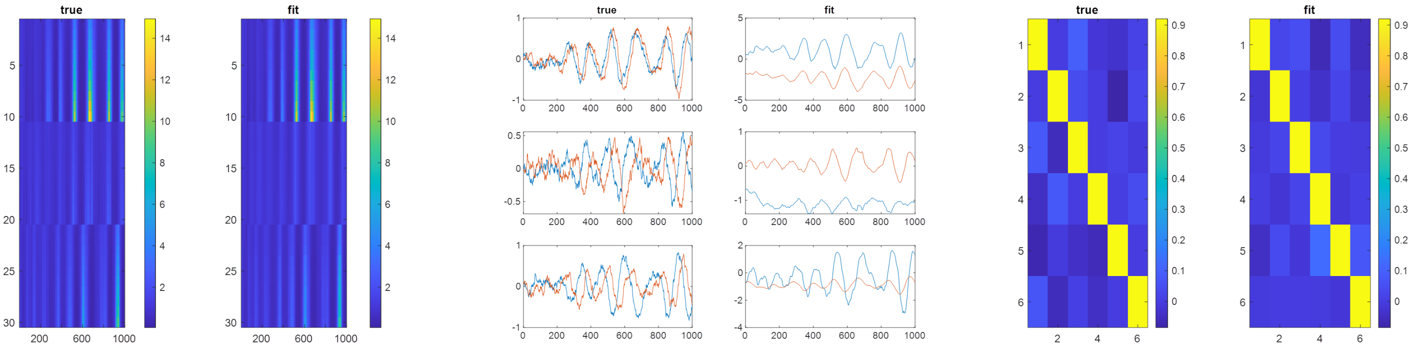
Figure 6: (Frobenius) norm for **d**, **C**, **b** and **A**

Some results are in Figure 7. These are means from iteration 1000 to 2000.

Personally, I think the fitting is not bad and the mixing is fine (in previous version, we may need 10000 iterations to get a not bad result. But here 2000 iterations did even better than before).

I also fit the model by assuming one cluster. Again, the fitting for overall mean firing rate is perfect. This is why it's important to make the loading, **d** and **C**, also be cluster-dependent (loading within clusters are correlated).

If the loading only depends on neuron index and will not change for different clustering assignments, it's impossible to do clustering (at least in this case),



(a) overall mean firing rate

(b) latents

(c) dynamics **A**

Figure 7: LDS sample with labels

since the loading is enough to capture all the patterns.

**Think more on it.** Maybe we can do MV-(Poisson)-GLM, to make all loading within cluster correlated.

### 5.1.2 Unlabeled Data: Clustering

To give the full path of clustering, I show results in GIFs. All results can be found in this folder. The code can be found in this folder.

Basically, in my current implementation, algorithms are good to merge clusters. However, generating new clusters is very hard... In other words, the newly generated cluster will usually not be sampled.

**This is a big problem**. This makes DPMM loses its power. **Fix that later**.

## 5.2 Simulation 2: Generate Latents, without Specifying Linear Dynamics

In this simulation, there are again 3 clusters with 2 latents and 10 neurons in each. Now, the latents are generated directly without specifying the underlying linear dynamics of latents. However, each latent is generated independently, so the linear dynamics matrix **A** should be roughly diagonal. The details of simulation can be found in the code.

### 5.2.1 Labeled Data: No Clustering

As in simulation 1, the data is fitted by the true cluster assignment (3 clusters) and forcing all neurons belong to single cluster. The code can be found here.

This time I ran 1000 iterations. The trace of the samples is shown in Figure 8.

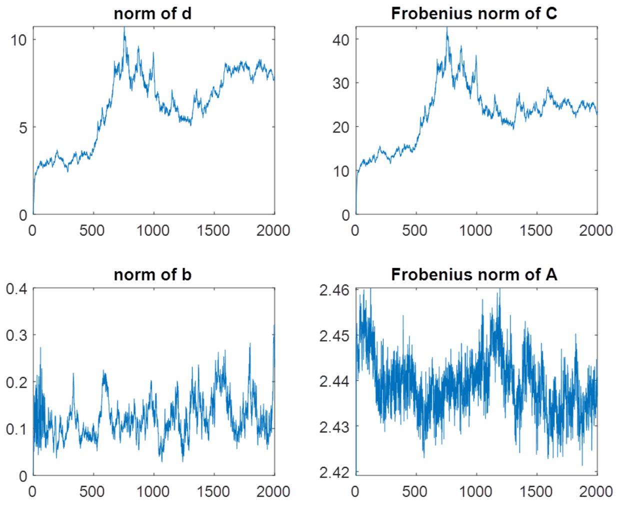
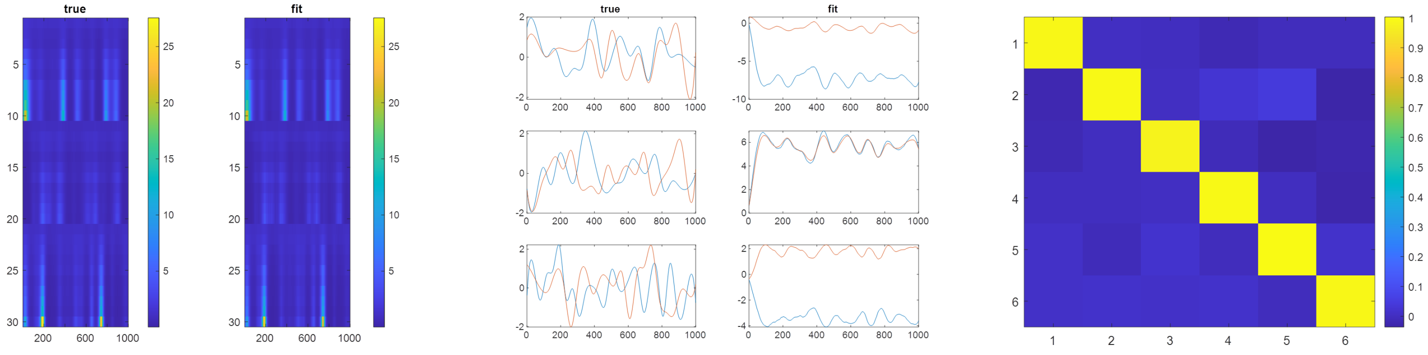


Figure 8: (Frobenius) norm for **d**, **C**, **b** and **A**

Some results are in Figure 9. These are means from iteration 500 to 1000.



(a) overall mean firing rate

(b) latents

(c) dynamics **A**

Figure 9: LDS sample with labels

This time the mixing of loading (**d** and **C**) is not fine. That reminds me that **I may need to make loading within cluster related by MV-(Poisson)-GLM directly.**

#### 5.2.2 Unlabeled Data: Clustering

Again, all results can be found in this folder. The code can be found in this folder.

Besides fitting 3 clusters with 10 neurons each, I further fit a larger scale example (50 neurons for each cluster).

# 6 TODO

(1) Think about MV-(Poisson)-GLM. Check ref1 and ref2.

(2) After resolving (1), see if the MCMC is still trapped in local optima sometimes.

(3) Find a more efficient way to generate new cluster parameters, otherwise the newly generated ones will always be rejected.

(4) improve DPMM. In current brute force implementation, the number of potential clusters can even go beyond number of neurons ($N$). There are several improvements, e.g. Kalli et al., 2011, Ge et al., 2015 and Hastie et al, 2015. Check them later.

(5) After all of them are resolved, switch to GP version