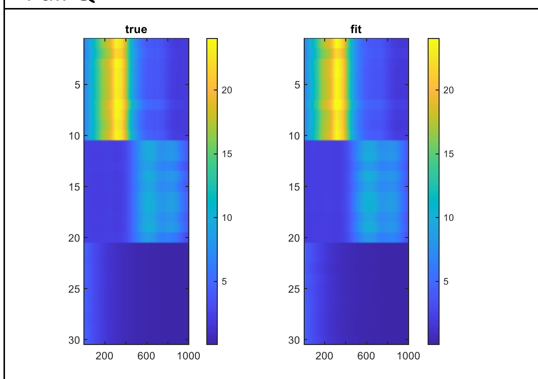
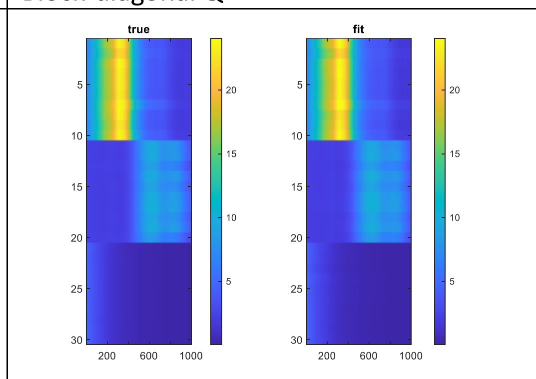
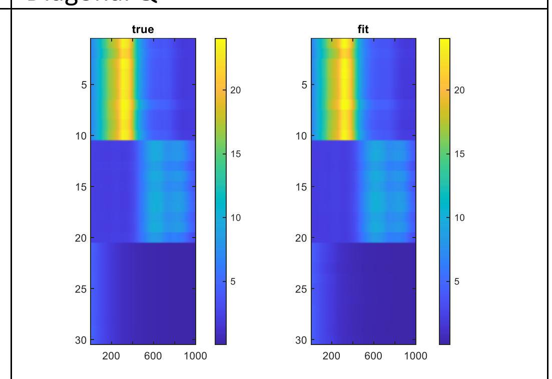
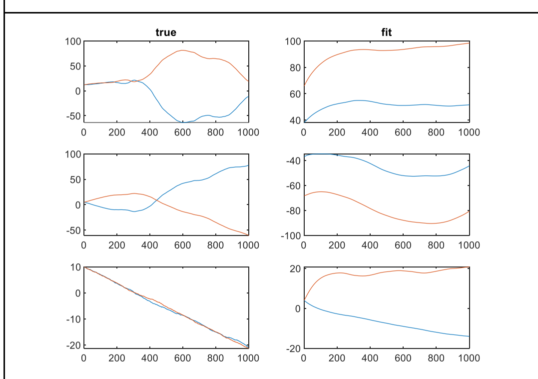
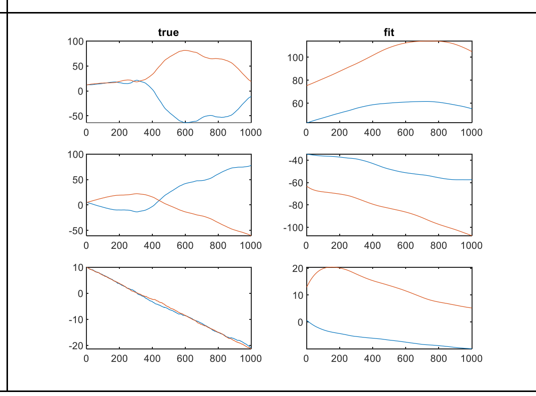
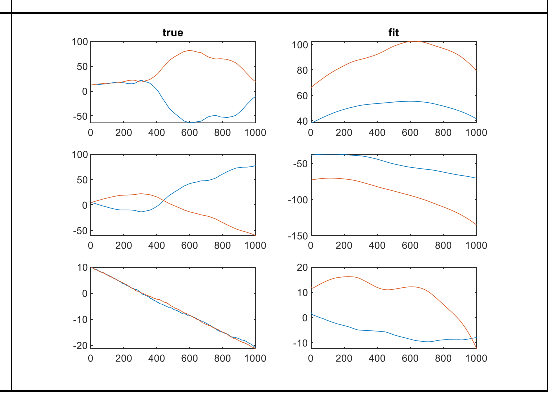
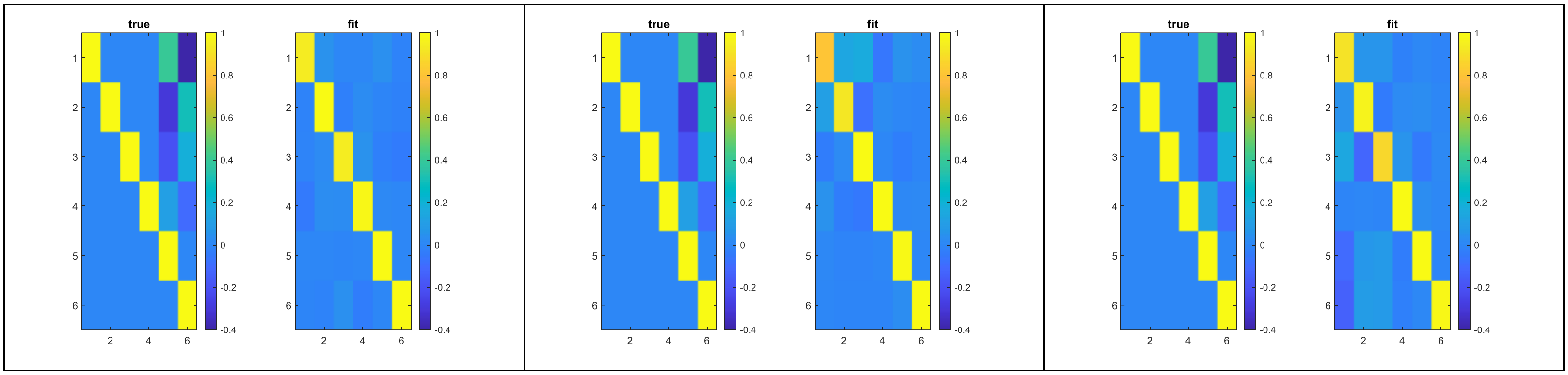
Labeled Data, Cluster-invariant Loading

Before doing clustering, I need to see if the model fitting for labeled data is fine. See the models and fitting details in [MCMC_LDS_v3.pdf]. In current model, the loading, i.e. d_i and $\boldsymbol{c}_i$ in $\log \lambda_{it} = d_i + \boldsymbol{c}_i' \boldsymbol{x}_t^{(z_i)}$, are cluster-independent. They only depends on the index of observation: even the cluster assignment of neuron i changes, the loading will not change.

Here, I show the fitting results for (1) no constraints on Q (code: sim_LDS_noConstraint.m), (2) block-diagonal Q (code: sim_LDS_blockDiag.m) and (3) diagonal Q (code: sim_LDS_diag.m). In the simulation, there are 3 clusters, with 10 neurons in each cluster. The number of recording step is 1000. 2 latent vectors for each cluster.

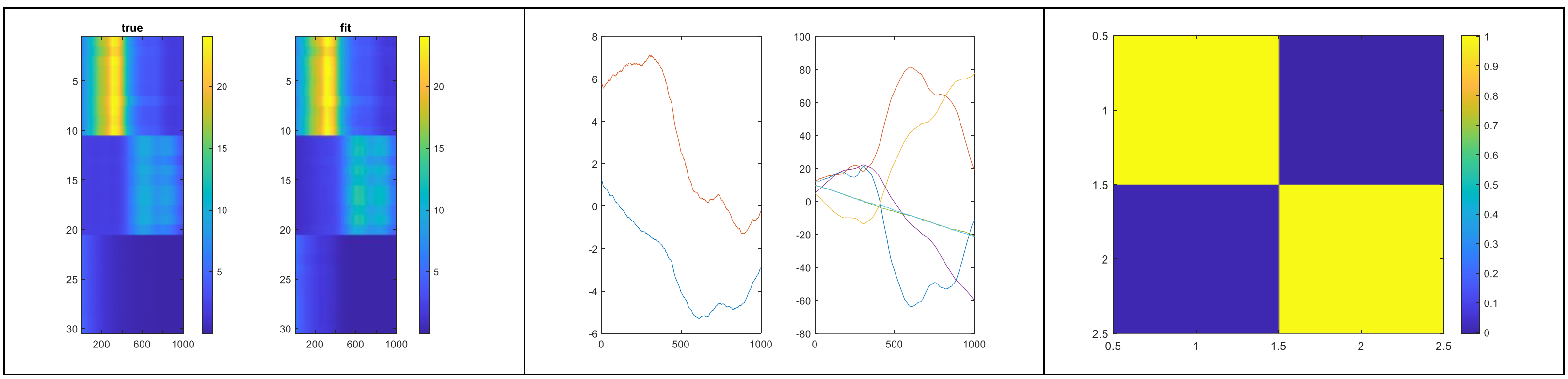
The results are mean of iteration from 50 to 100. From top to bottom, there are 3 fitting results: (1) overall mean firing rate for 30 neurons with 1000 steps; (2) the latent vectors for each cluster and (3) the linear dynamic matrix A to show inner- and inter-population relationships.

Full Q	Block-diagonal Q	Diagonal Q
		
		



Since the dimension of latents is not large, the speed for these 3 are nearly the same.

The overall fitting for mean firing rate is perfect, but the pattern for the latents and linear dynamics are not OK. That means the simulation is bad: it’s unidentifiable. I further force all observations belong to single cluster, with block-diagonal Q model. The results are here:



The overall fitting are still perfect. **That means the pattern can be purely captured by cluster-independent loading.** This will ruin the clustering. Maybe I need to find a better simulation example such that everything is identifiable.

Unlabeled Data, with Known Loading

Before fix the problem caused by cluster-invariant loading, I first tried to do clustering based on known loading d_i and $\boldsymbol{c}_i$. If this works fine, the remaining problem is just loading-related issue.

I use the same simulation setting and do clustering with both MM (LDS_MM_demo.m) and DP (LDS_DP_demo.m). The details of MM and DP wrappers can be found in cluster_XXX.pdf. The four results are shown in gifs:

(1) MM, start from 1 cluster, number of cluster (K) = 4: noLoading_MM_below.gif
(2) MM, start from N clusters (each neuron form a single cluster, K = N): noLoading_MM_above.gif
(3) DP, start from 1 cluster, alpha = 1: noLoading_DP_below.gif
(4) DP, start from N clusters, alpha = 1: noLoading_DP_above.gif

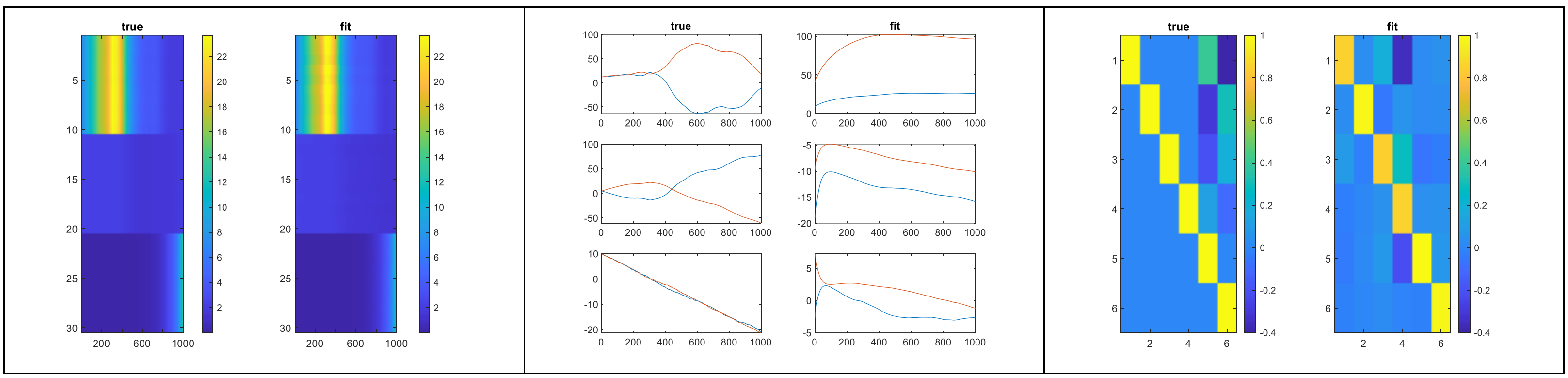
Everything works fine. So let's deal with loading.

DETOUR: Newton-Raphson vs. Adaptive Smoother

Before dealing with loading, I first tried what's the difference between estimations from Newton and Adaptive smoother for latent vectors ($\boldsymbol{x}_t$). Here I show 2 examples (newtonTest.m).

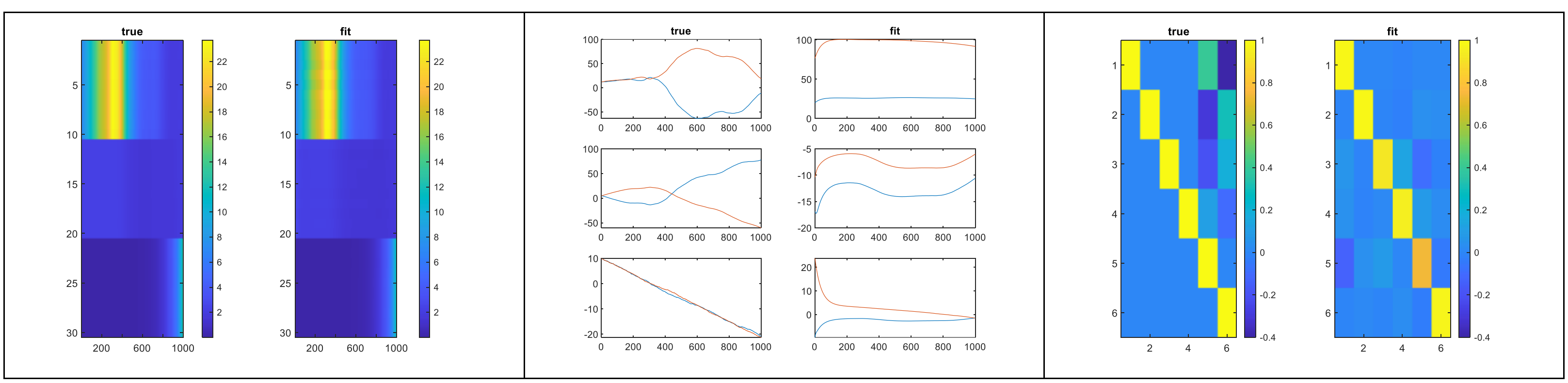
Labeled Data, Cluster-dependent Loading

To make the loading d_i and $\boldsymbol{c}_i$ depend on cluster assignment, I add one more layer to distribution of loading, as shown in MCMC_LDS_v4.pdf. The simulation results are here. The only difference is that the loadings are generated based on cluster assignment. Se code in sim_LDS_blockDiag_v2.m. Here I just use the block-diagonal Q version. The results are means of iteration 50 to 100.



Well, again the overall fitting is perfect again, but others are not very good. That suggests the unidentifiable simulation.

I further tried to use NR for latent vectors ($\boldsymbol{x}_t$) (without warm start from adaptive smoother yet. Improve that later…)



Hmmm, although the overall results are similar, the latents (2 and 3) and dynamics are kind of different. Let's further see differences of Q estimation:

True Q	Q from NR	Q from adaptive smoother

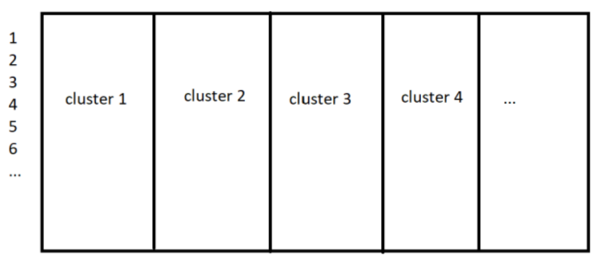
1.0e-03 *						1.0e-03 *						1.0e-03 *					
0.0100	0	0	0	0	0	0.0699	-0.0090	0	0	0	0	0.1883	-0.0302	0	0	0	0
0	0.0100	0	0	0	0	-0.0090	0.0675	0	0	0	0	-0.0302	0.0689	0	0	0	0
0	0	0.1000	0	0	0	0	0	0.1516	0.0377	0	0	0	0	0.1728	0.0377	0	0
0	0	0	0.1000	0	0	0	0	0.0377	0.0464	0	0	0	0	0.0377	0.0403	0	0
0	0	0	0	1.0000	0	0	0	0	0	0.1163	-0.0063	0	0	0	0	0.1061	0.0035
0	0	0	0	0	1.0000	0	0	0	0	-0.0063	0.0546	0	0	0	0	0.0035	0.0550

Estimation from NR seems a bit (just a bit) better…

Anyway, let's see what will happen when using the cluster-dependent loading for clustering.

Unlabeled Data, with Unknown Cluster-dependent Loading

Now, the loading is expanded for different categories. Take $\boldsymbol{d} = (d_1, \ldots d_N)'$ for example, it is expanded to a N-by-J matrix, where J is the current max number of cluster.



If there exists observation at row i and column j, estimate the value.

If there's no observation for row i and column j, but there are some observations in cluster j (can estimate $\mu_d^{(j)}, \sigma_d^{2(j)}$), sample data from $N(\mu_d^{(j)}, \sigma_d^{2(j)})$.

If there's even no observation in cluster j, just generate things by prior.

I just use MM with K = 4 and start from single cluster. The results: loading_MM_below.gif.

Well, I have to say the algorithm is unrobust now. Sometimes, the adaptive filtering will fail and sometimes the Newton-Raphson will fail. But this simulation just shows some potentials…

TODO

(1) More meaningful & identifiable simulation.
(2) See if I can get a warm start for NR in loading update.
(3) Combine adaptive smoother and NR if necessary.