# V1 data: Model Comparison

Neuron = 13, nKnots = 7 for both X and G

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Adaptive CMP (smoothing) | Adaptive Poisson (smoothing) | Static CMP | Static Poisson |
| Q | 1e-4\*(0.313, 0.118, 0.169,  0.176) | 1e-4\*(0.258, 0.210) | -- | -- |
| Window size | 10 |  |  |  |
| Training llhd | -1.024\*1e4 | -1.056\*1e4 | -1.056\*1e4 | -1.081\*1e4 |
| Held-out llhd | -1.059\*1e4 | -1.080\*1e4 | -1.065\*1e4 | -1.093\*1e4 |

\* the held-out log-likelihood is still calculated by averaging the closest 3 parameters.

[~, sort\_id] = sort( |held\_out\_x(j) – training\_x|);

id = sort\_id(1:3);

theta = mean(theta\_fit(:, idx\_train(id)), 2);

The large differences of log-likelihood between training & held-out set may suggest the parameters are not stationary within a trial.

Plot the trace of estimated mean, with observation in background:

|  |  |
| --- | --- |
| Training set | Held-out set |
|  |  |

Non-surprisingly, adaptive CMP can capture more details (sensitive to trial-to-trial variation).

# Hippocampus data

Neuron = 73, nknots = 4 for both X and G. Here, I use non-circular basis, just because I didn’t successfully coding the circular basis...

X = getCubicBSplineBasis((pos - min(pos))/range(pos),nknots,false);

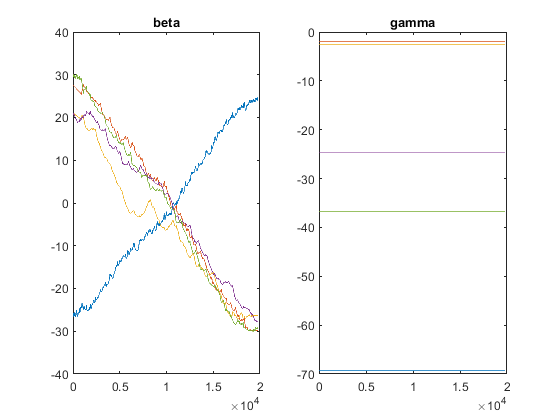
Observations (sliding window = 100):



Model fit:

Optimized Q = 1e-3\*(0.884, 1.000, 0.500, 0.500). window size = 20

Red = fit; blue = observation



Well, since the observation is so sparse. The lambda is always < 1 and constantly (The same for all other examples I tried). That doesn’t make sense… Maybe it’s better to use a coarse bin?

Here I re-bin the data as follows:

bin = 10;

pos = zeros(ceil(length(pos\_raw)/bin), 1);

spk = zeros(ceil(length(pos\_raw)/bin), 1);

for k = 1:ceil(length(pos\_raw)/bin)

raw\_idx = (bin\*(k-1) + 1):min((bin\*k), length(pos\_raw));

pos(k) = mean(pos\_raw(raw\_idx));

spk(k) = sum(spk\_raw(raw\_idx));

end

neuron = 10

Observations (sliding window = 10):





Since the spiking is too sparse, window-mean is very different from bin spike count.

OK, let’s fit.

nknots = 4 for both X and G.

Qopt =

1.0e-03 \*

0.0041 0.0580 0.9990 0.9993

Use window size = 50



Also plot (1) left: spike counts (blue) & CMP mean (red); (2) right: smoothing FF (blue) & model FF (red).



OK, that looks much better.

# Q tune & window size

Yes, I do agree tuning Q is best and the model is not very sensitive to window size. However, I’m just kind of “scared” by the case 2 & 3 previously.

## Q tune issue

The annoying case 2: the optimized Q is too small, even a super large window size cannot remedy it.



|  |  |
| --- | --- |
| No Q tune: Q = diag([1e-3 1e-3]), k = 5 (optimized window size for filtering likelihood) | No Q tune: Q = diag([1e-3 1e-3]), k = 15 |
|  |  |
| No Q tune: Q = diag([1e-3 1e-3]), k = 100 | Q tune: Q = diag([2.5e-6 2.4e-7]), k = 400 |
|  |  |

So, it’s best to check the log-likelihood against window size roughly. If we need a very large window size (or the plot looks weird), then that may be a sign of problems.

## Window Size

Case 3:

|  |  |
| --- | --- |
| Window size (k) = 10 | Window size (k) = 100 |
|  |  |

Maybe it doesn’t harm to check log-likelihood. In case 3 k = 100 is bad, but in case 2 k = 100 is great. One more thing is that super large window size takes long time to run, maybe because of bad coding…