Hi all,

Here is an updated email.

I have been sorting lots of spikes recently, and wanted to share with you some updates/comments/thoughts I have regarding both the detection process and the sorting process.

\* After I finished to write this mail I saw it got quite long, I hope it is well organized and make some sense ... :)

**Spikes detection:**

**1. filtering:**

-The old filtering we used in the lab introduced the fast ringing effects (around 5kHz). So those "artifacts" you used to see are no more the a result of the sharp high cutoff filter. To overcome this simply use a high-pass filter (a 600-Hz high-pass) instead of a band-pass filter (i.e. do NOT do a 6-kHz lowpass; this is not a problem as the neurologger does a 7-kHz lowpass anyway).

- With this new filter, the spikes are (i) cleaner; (ii) a bit larger in amplitude; and (iii) a bit narrower.

- In terms of SNR it seems to have a slight increase - both signal and noise are increased, but signal increases more (as it has more power in the high freq than the noise).

- This new filtering also requires some changes in the library (see next), as it CAN create narrower spike shape. This is mostly required for narrow interneurons that now get even more narrow, and are anyway underrepresented in the library.

**2. Library of acceptable spikes shape:**

- I added 3 spikes waveforms from 3 putative interneurons (highly isolated!!). These cells had very narrow spikes, and thus had low correlations with the old library.

- I also checked how the new filtering affects the correlations with the library for several putative pyramidal spikes, and it seems there is no much effect there, because (i) width-wise, there is high representation for various middle-width waveforms, and (ii) the high-freq noise doesn't change much in terms of correlations (and anyway the library was built with waveforms *without* the ringing "noise"...).

- I think we should lower the threshold for the correlations with the library. I used a threshold of **r = 0.8** for all of my recordings. I did that because I had several examples, right at the beginning of the sorting, that I had good cluster which lost ~50% of its spikes due to the high correlation threshold of 0.9, and I didn't want to have many cases that I need to resort as only AFTER sorting I can determine whether a cluster spikes were lost due to high thr (trimmed histogram). I also added code to plot the histograms of correlations to the library for all units in a given NTT file – see example file attached (*'spikes\_b2289*…') – grab me to show you the code for this; and I added an option to export the waveforms that were discarded in the detection process due to low lib corr (they appear as a unit in the NTT file) - see next (code improvements).

**3. Code improvements:**

Grab me to show you the code improvements. Here is a short summary of what I improved:

- I moved the re-referencing to the detection stage (I removed it from the nlg2nlx converter). This was done for 2 reasons: (i) we might want to have different re-referencing for spikes detection and for LFP; and (ii) it enables to use the reference tt as an additional tt in the coincidence detection step (see below).

- I implemented a coincidence detection (CD) code that runs much faster (only couple of seconds instead of the current several minutes for-loop).

- CD now detects both positive and negative events (as we shouldn't necessarily assume the noise to be specifically positive).

- I separated the CD thr from the spikes thr levels.

- new param: *CD\_n\_TT\_thr*, enables choosing how many tetrodes should have simultaneous events such that we consider that as a CD event. Currently I used here the number of *connected* tetrodes (including the reference!! I do *not* re-reference the ref tt, and look also for simultaneous event on the ref tt). This will be more relevant for the 16-TT drive of course, where we should use a lower value than the total number of tetrodes (16), probably somewhere between 5 to 10 tetrodes. With higher number of tt’s the probability for a non-real (false) CD event is much lower, so we expect to have much less false alarms here. But with the 4-tetrode drive, if you have *less* than 4 connected tetrodes (i.e. *CD\_n\_TT\_thr* < 4), it may cause a problem - This was actually a problem in one of my bats: I had only two tetrodes connected, so the probability to have a CD event by chance was relatively high, and in some days we initially lost almost ~50% of the spikes in some clusters. But there is in fact a solution for that problem - how to detect and what to do in such a case - see the point below regarding saving to the ntt file all the detected spikes (including the ones removed by lib/duplicates/CD). Grab me to explain you in more detail about this.

- In addition to the CD window (window for coincidence-detection), there is now also an optional invalidation window: i.e. when there is a noise detection (CD), we can choose also to invalidate the surroundings of the event. I added this option as many times we have noise which lasts a few msec or more but its amplitude crosses the thr only for a short duration: a dacaying ringing for example, which occurs very often, and where we want to invalidate a larger invalidation-window.

- I added figures that help visualizing the detection process: plotting the waveforms from the different spikes detection steps (library, duplicates, CD), both in waveform and clusters view. See attached 2 figures *spikes\_b0148*…, which show an example that demonstrates how we can quickly see that real spikes from a good cluster were lost in the CD process. Let’s focus on the clusters plot: The first row shows the valid spikes, and the next 3 rows are the spikes that were removed based on library (2’nd row), based on duplicates = minimal separation (3’rd row), or based on coincidence detection (4’th row). I encircled a specific good cluster in the 1’st row, but you use that we also lost a lot of spikes from this cluster based on coincidence detection – encircled in the 4’th row ! This shows you that the coincidence detection based on just 2 tetrodes in this case (bat 0148) removed lots of good spikes ! The next point addresses how to overcome this problem.

- I added an option to save an NTT file that contains all the detected waveforms, including the ones removed by lib/duplicates/CD. The valid spikes are assigned to unit 0 (“unsorted”), and the invalid spikes were assigned to units 1,2,3 , respectively. This way I can see (in a spike-sorting program) which waveforms were rejected in each step, and moreover, I can validate back spikes that look as belonging to a valid cluster – for example, in the attached clusters plot, I can merge back the spikes from the encircled cluster in 4’th row to the spikes in the cluster in the 1’st row.

- I corrected nlx headers for ntt and ncs files! I finally managed to find the correct way to write these headers, so the neuraview and plexon will open the files correctly without warnings, and will display the voltage and time values correctly :) Grab me to show you how to do this.

- I added a function to convert plx files to ntt. I use this function only when I use the invalidate waveforms option in plexon (see using plexon below).

**4. detection params:**

- Spike detection thr: instead of using spike detection thr of 5 \* median(abs(signal)) , I used 6 or 7. I found that the threshold of 5 \* median was too low for most of my data – we looked with Nachum at many examples and reached the conclusion that a threshold of 5 is too low indeed – and so I used a threshold or 6 in about half of the cases and 7 in the other half of the cases. Usually the choice of 6 or 7 was consistent per tetrode, for all days of that tetrode – it seems to have to do with impedance/noise levels per tetrode.

- CD threshold: for the CD thr I used 6 \* median(abs(signal)).

- CD window (what timelag between tetrodes events is considered as coincidence detection): We should consider decreasing this values substantially. For most of my preprocessing I used 1 msec max lag = 32 samples, but after looking at this together with Shir and Nachum, we think that we should reduce this number to 4 or 8 samples. We need to explore this point more – although it should be noted that for 16 tetrodes the exact choice of this parameter will be less critical, because whatever CD window you will choose, it’s highly unlikely to have 5 or 10 tetrodes cross by chance the CD threshold within that CD-time-window – unless this is a real noise/artifact that you truly want to remove...

- Duplicates removal - *min\_win\_seperation*: I used here 24 samples (750 usec). We definitely loose some spikes here but I think it just makes the sorting much easier.... Though, note the next section about plexon being able to cope with that.

**5. using plexon:**

- Please grab me (or Shir) to show you how to use Plexon. We both used it now for several months, so we have lots of experience with it now…

- Time dynamics: It is simply a MUST!!! For most of the days that I sorted, the sorting was almost hopeless without this feature (if you care about good isolation...). It both enabled separating clusters that I would have never be able to cluster without the time segments, and it allowed me to understand that sometimes a "well-isolated cluster" in the pooled time view, actually contains two or three cells....

- Saving sorting results in plx format: Usually (when not using the *invalidate waveforms* option in plexon) you should simply save the sorted file as NTT (as it was loaded from an NTT file). But, in case you use the *invalidate waveforms* option, you should save it as plx file format – otherwise (saving as NTT) it will ignore the invalidation, and invalid spikes will be regarded as unsorted once you will load the file again. (\* when saving as plx you should make sure to uncheck '*export invalidated waveforms*').

- I've gathered some spike-sorting tips with Plexon - all kind of technical tricks that you can do with the plexon sorter. Anyone that plans to sort sometime is very welcome to ask for a practical demonstration.

On a general note: doing this sorting (with time segments) was VERY (!) time consuming. Just thinking to do such sorting for 16-TT recordings gives me a headache...

Automatic sorting algorithms are now trying to better deal with such temporal-drifts, and we should definitely give them a try. Specifically, the recently-released **kilosort2** was updated to handle such drifts (you can find [here](https://xmail.weizmann.ac.il/owa/redir.aspx?C=NIVw22j9BaOoXWIAjqoPKr7ePiD8rlvSf0oYdzIBwjcn1PwG8MPWCA..&URL=https%3a%2f%2fgithub.com%2fMouseLand%2fKilosort2%2fwiki%2f3.-More-on-drift-correction) some explanation about the algo). They state in this document that it should work also for tetrode data and not only for linear probes such as NeuroPixels, as the algorithm doesn't assume anything about the spatial arrangement of the contacts.

\*\*All the updated code is located in my [gitHub project](https://xmail.weizmann.ac.il/owa/redir.aspx?C=SRyeyCw_21JuqhRUrgR8lg7gr3oE4o-A3TaHooSzbkIn1PwG8MPWCA..&URL=https%3a%2f%2fgithub.com%2fTamirEliav%2fLargeScale).

Tamir

D:\Tamir\batlab\batlab_shared\matlab\pre_processing_of_data\spike_detection_and_library\spikes_b0148_d170627_TT4_artifacts_clusters.tifD:\Tamir\batlab\batlab_shared\matlab\pre_processing_of_data\spike_detection_and_library\spikes_b0148_d170627_TT4_artifacts_waveforms.tif

L:\Analysis\pre_proc\2289\20180518\spikes_sorting\spikes_b2289_d180518_TT1_lib_corr_by_units.tif