LTP analysis cheat sheet

1. Load the abf files for base, chase, induction, and where possible, cross-talk

NOTE: in case the Clampex protocol consisted of several runs per trial, the block's segments (or sweeps) should contain the minute average responses in two sweeps (one per stimulation pathway); otherwise, the block's segments contain immediate (or raw) synaptic response data and you may need to average these (more about this later)

WARNING: Check that data has:

1. appropriate names e.g. `base_X`, `chase_X`, `xtalk_X`, `tbp_0_X` etc
2. appropriate structure:

- for evoked epsc (or epsp), `base_*` and `chase_*` are `neo.Block` objects expected to contain and *EVEN* number of segments: with the *EVEN* indices(0, 2, 4, etc) containing signals related to one pathway index (say, path 0) and the *ODD* indices (1,3,5, etc) containing signals related to the other pathway index (path 1)
- make sure the `xtalk_*` blocks contain cross-talk data (if recorded)
- make sure that you can figure out to which pathway the LTP induction was applied

2. Perform analysis

2.1 Collect *Test* and *Control* pathway responses as separate blocks

Collect the blocks in two separate lists, e.g. `baseline_blocks` and `chase_blocks`; make sure that: * each block in the list has exactly TWO segments (minute-averages, one per pathway); * you DO NOT include cross-talk!

Concatenate the sweeps (i.e., `segments`) corresponding to each pathway in separate blocks, e.g.:

```
path0_baseline = neoutils.concatenate_blocks(baseline_blocks, segments = 0, analogsignals = [0,1,2], \
    name = result_name_prefix + "_path0_baseline")
path0_chase = neoutils.concatenate_blocks(chase_blocks, segments = 0, analogsignals = [0,1,2], \
    name = result_name_prefix + "_path0_baseline")
path1_baseline = neoutils.concatenate_blocks(baseline_blocks, segments = 1, analogsignals = [0,1,2], \
    name = result_name_prefix + "_path1_baseline")
path1_chase = neoutils.concatenate_blocks(chase_blocks, segments = 1, analogsignals = [0,1,2], \
    name = result_name_prefix + "_path1_baseline")
```

The `analogsignals` parameter indicates which signals you want to keep; these can be integers (signal index) or strings (signal names) in which case make sure all signals are named similarly across the data; at a minimum, you should include the membrane current signal and the command voltage.

You should end up with four blocks: `path 0 baseline`, `path 0 chase`, `path 1 baseline`, and `path 1 chase`; **NOTE:** Write down which of the two paths is the ***Test*** pathway !!!

2.2 Set up cursors manually:

Open each concatenated blocks in SignalViewer, select the membrane current axis (containing synaptic responses), and place vertical cursors in that axis (do NOT select multi-axis cursors!)

The cursors names, window width and placement are given below. both window widths and X position can be readjusted. In particular the X position of the *peak* cursors should be consistent across the sweeps. ***NOTE:*** names are case-sensitive, and must be set EXACTLY as shown here.

For single-pulse stimulation set EXACTLY 5 (five) cursors

- Rbase (0.01) $\mapsto$ current baseline BEFORE deplarization transient (for membrane Rs and Rin)
- Rs (0.003) $\mapsto$ peak of the first capacitive transient
- Rin (0.01) $\mapsto$ steady-state current during depolarization
- EPSC0Base (0.01) $\mapsto$ current baseline BEFORE 1st stimulus artifact (1st pulse)
- EPSC0Peak (0.005) $\mapsto$ trough of the 1st EPSC

For paired-pulse stimulation set EXACTLY 7 (seven) cursors: five as above ***plus*** the following two

- EPSC1Base (0.01) ↦ current baseline BEFORE 2nd stimulus artifact (2nd pulse)
- ESPC1Peak (0.005) ↦ trough of the 2nd EPSC

2.3 Create epochs:

In the signal viewer, select the axis containing the cursors - you can simply click on the axis; optionally, use the axis selector widget to show only the axis plotting the synaptic responses.

From the signal viewer `Epochs` menu, select `Make Epochs in Data/From Cursors`

- in the dialog, select all cursors shown, press `OK`
- in the next dialog, *set the epoch name to "ltp"* (lower case)
- make sure the following checkboxes are selected:
    - `Embed in all segments`
    - `Relative to each segment start`
    - `Overwrite existing epochs`
- press `OK`

You should now have exactly *one* epoch with 5 or 7 intervals in each segment. **NOTE:** the epochs are *embedded* in the data - hence, they will be saved with your data on disk (in a pickle file).

Repeat this step for all four blocks. **NOTE:** When the `Baseline` and `Chase` data are *from the same pathway* have the same time base, the cursors will already be visible, but you may need to readjust their positions before creating an epoch in the block you are visualizing.

2.4 Analyse each pathway

```
result_path0 = ltp.analyse_LTP_in_pathway(path0_baseline, path0_chase, 0, 0, \
    signal_index_Vm = 1, normalize=True)
result_path1 = ltp.analyse_LTP_in_pathway(path1_baseline, path1_chase, 0, 1, \
    signal_index_Vm = 1, normalize=True)
```

The arguments are:

- baseline block
- chase block
- index of the membrane current signal (where the EPSCs are recorded)
- index of the pathway (e.g., 0 or 1)
- keyword arguments indicate the membrane voltage signal (~ the command signal) and whether to normalize the amplitudes

The result is a `pandas.DataFrame`; you can save it as a `pickle` and export it to `CSV`. Row indices indicate the minute of recording (row index 0 indicates the first minute AFTER LTP induction). Columns are as follows:

- Rs (MΩ) - series resistance
- Rin (MΩ) - input resistance
- DC (pA) - baseline membrane current
- EPSC0 (pA) - amplitude of the first EPSC
- EPSC1 (pA) - amplitude of the second EPSC
- PPR - pair-pulse ratio (EPSC1/EPSC0)
- ISI (s) - inter-stimulus interval (currently not calculated - all NaNs)
- EPSC0Norm - EPSC0 amplitude normalized to the average EPSC0 over last 5 min of baseline (by default)
- EPSC1Norm - EPSC1 amplitude normalized to the average EPSC1 over last 5 min of baseline (by default)

To plot any of the columns call, e.g.:

```
plt.plot(result_path0.index, result_path0.EPSC0Norm, 'o')
```

2023-05-10