About LTP analysis

Preamble

This file describes the workflow for analysis of LTP experiments (voltage-clamp) involving the alternative (i.e., interleaved) stimulation of two pathways.

Each pathway is stimulated at 0.1 Hz, using paired pulses. This means that during a minute, six paired-pulse stimuations will be delivered to each pathway.

The paired pulse stimulation consists of two pulses delivered at short (50 ms) interval to the *same* pathway via the stimulus isolation unit and stimulus electrode.

Hardware and software

The document is specific for the following devices & software:

Hardware: *Amplifier*: MultiClamp 700 B; *signal acqusition* device: Digidata >= 1440

Software: `pClamp 11` software suite (data acquisition: `Clampex`)

Interleaved stimulation protocol

In `Clampex`, interleaved two-pathway stimulation can be achieved by setting up a protocol in `Episodic Stimulation` mode (selected in the `Mode/Rate` tab of the `Protocol editor` dialog).

The protocol consists of a *trial* with six *runs* per trial, and two *sweeps* per run; alternative waveforms and alternative digital configuration are *enabled*.

Each *sweep* lasts one second, and within each run the sweeps are recorded with five seconds interval (*start-to-start*).

Within a *trial*, there are six *runs* at every 10 seconds interval (*start-to-start*). The paired-pulse stimulation should be placed earlier in the sweep, but see below.

The stimulations are *interleaved*, meaning that in every one minute *trial*, each pathway is activated six times in a minute, with the average synaptic responses for each minute recorded and stored in abf files.

NOTE: Make sure the protocol records the following signals:

- the membrane current signal (this contains the synaptic responses !)
- the command voltage signal (usually this is the secondary output from the amplifier in voltage-clamp mode, but check the specification of your amplifier)
- the signals carrying the digital triggers for the stimulation electrodes to *both pathways*.

In `Clampex`, all these signals should be configured in the program's `Lab Bench` - make sure they are linked to your amplifier via the `Telegraph` option.

It is recommended for these signals to be given meaningful alias names e.g. in `Clampex`'s `Lab Bench`.

For example, the analog input 0 (by default named "IN0" in `Clampex`) provided it is physically connected to the primary output of the amplifier channel 0, can be also named as:

- "Im_prim_0", to be used as the "primary output signal in *voltage-clamp*, for the amplifier channel 0"
- "Vm_prim_0", to be used as the "primary output signal in current-clamp, for the amplifier channel 0"

Likewise, the analog input 1 (by default named "IN1" in `Clampex`) provided the analog input 1 of the digital acquisition box is connected to the secondary output of the amplifier channel 0, can be also named as:

- "Vm_sec_0", to be used as the "secondary output signal in *voltage-clamp*, for the ampltifier channel 0"
- "Im_sec_0", to be used as the "secondary output signal in *current-clamp*, for the ampltifier channel 0"

... and so on ...

NOTE: The trigger signals are typically output from the digital acquisition device as TTL signals, and fed into the stimulus isolation units as `external triggers`. With the use of BNC splitters (or `tee`) these signals can be fed back to analog input ports of the digital acquistion box so that they can be recorded during the experiment. In `Clampex` these input signals should be condigured in `Lab Bench` and given a meaningful alias (or name) e.g. `Stim_0`, and `Stim_1`.

Sweep waveforms

In `Clampex`, the paired-pulses stimulation is set up using the `Waveform` tabs in `Protocol editor`

Within each *sweep*, there should be a small depolarizing step at a suitable delay form the stimuli, necessary to monitor (or calculate, offline) the series and input resisatances. In `Clampex`, this depolarizing step is set up using the `Waveforms` tab of the protocol editor, and should be placed either near to the beginning (typically) or the end of the sweep.

With this protocol, `Clampex` will write the average *trial* data - i.e., averaged over a minute period - to a file containing two *sweeps* (each being the average of the six sweeps with the same index from each *run*).

To record for several minutes (or indefinitely) in `Clampex`, you also need to set up a *sequencing key* to record using the protocol set up as above then automatically calls the same protocol again - i.e., the same *sequencing key - with a 10 s interval*.

In this way you will end up with a series of `abf` files, each containing two *sweeps*, each with separate minute-average data from the two pathways.

After an initial 10-12 minutes of recording ("baseline") an LTP induction protocol is applied to one of the pathways (usually in current-clamp, but that depends on the induction protocol). This should also be recorded, and the recommended practice is that the name of the file should reflect the pathway to which the LTP induction was applied.

After LTP induction, the interleaved stimulation protocol is applied, recording pathway-specific minute-average data indefinitely (or as long as the cell survives).

The analysis is currently done offline, and starts by splicing then concatenating the baseline and chase sweeps for each pathway into four data blocks.

The EPSC amplitudes are then calculated based on epochs set up in each of the foour data blocks, by way of cursors created in `Scipyen`'s `SignalViewer` window.

Analysis steps

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 the 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 which pathway the LTP induction was applied
to:
        - e.g., check which digital signal was used inside the `tbp_*` blocks, unless their names already re

```

2. Load or generate LTP analysis options; this is a `dict` so it can be modifed at the console.

If you do not yet have an `LTPOptions` `dict` saved somewhere, call:

```
LTPOptions=ltp.load_synaptic_plasticity_options()

```

otherwise, just load it (usually, from a pickle `*.pkl` file)

3. Concatenate the blocks to generate baseline and chase data for each pathway

```
ltp_data = ltp.generate_minute_average_data_for_LTP(base, chase, LTPOptions, test_index, cell_name)
```

where:

`base:str` is a search string containing the common prefix for the baseline data, as a global pattern, e.g. `base_*`; make sure you include the asterisk and optional underscores so that you don't take into account other data objects named like `base*`, etc.

`chase:str` is a search string containing the common prefix for the chase data - see base for details

`LTPOptions:dict` is the object containing LTP analysis options , see point (2) above

`test_index:int` is the index of the test pathway (*either* `0` or `1`); **NOTE**: there may be more than two pathways stimulated in your experiment - just make sure this is set up so that the LTP induction is done on either pathway `0` or `1`.

`cell_name`: `str` - the name of your cell (for record-keeping)

The call above returns a `dict` containing:

- `Control`:`dict` with data for the `control` pathway:
    - `Baseline`: `neo.Block` with synaptic responses on `control` pathway *before* LTP induction
    - `Chase`: `neo.Block` with synaptic responses on `control` pathway *after* LTP induction
    - `Path`:`int` the index of the `control` pathway (*either* `0` or `1`)
- `Test`:`dict` with data for the `test` pathway - same structure as the `Control`, shown above.
- `LTPOptions`:`dict` a copy of the LTP options object passed as argument (see above)

4. View the `ltp_data` (double-click in the `User Variables` table to open it in a `DataViewer`).

Expand the tree and right-click on, say, the `Control/Baseline` then select `View` to view this in a `SignalViewer` window.

Navigate across thre sweeps (or segments) - use the bottom navigation widget - to make sure all data is right.

Make note of the following:

- name of the signal viewer window (last bit of the window tiytle, after the dash).
- index of the path you are viewing (e.g., `0` for `control` and `1` for `test`, or *vice-versa*)

5. Set up cursors and epochs.

Below, we assuming you are viewing the `Control/Baseline` data (both baseline and chase data of *the same pathway* ought to have the same time base or domain).

Set up the cursors

Call

```
ltp.setupLTPCursors(viewer, LTPOptions, pathway, axis)
```

where

`viewer:SignalViewer` is the Signal viewer object

`LTPOptions:dict` is your LTP options dict

`pathway:int` is the index of the pathway you are viewing, taken from the `ltp_data` object

`axis:int` is the index of the axis ofthe signal viewer, where the synaptic response is plotted.

- You should now see a set of 7 vertical cursors in the axis that plots the synaptic responses.

- You can drag their labels vertically, to stagger them (easier to read!)
- Also, in the signal viewer `Cursors` menu is a good idea to enable "`Cursors show value`"
- Make any necessary adjustment to the cursor's horizontal coordinates, buy dragging their lines.

NOTE: these cursors are tailored for a voltage-clamp experiment (hence recording synaptic currents) with paired-pulse stimulation (hence expecting *two EPSC* waveforms). In addition, a short, small depolarization at the beginning of the sweep is expected, to monitor the access (series) and input resistances.

The first three cursors relate to the calculation of the series (access) resistance and membrane (input) resistance - hence they are named accordingly:

- `Rbase` - baseline membrane current;
- `Rs` - series resistance - should sit on top of the peak of the first capacitive transient
- `Rin` - input resistance - should sit on the steady-state part of the wavveform, just before the repolarization

The last four cursors mark, respectively, the baseline (pre-simulation) and the peak of each of the two EPSCs, so they are named `EPSC0Base`, `EPSC0Peak`, `EPSC1Base` and `EPSC1Peak`.

Adjust the cursors' horizontal positions and window size

Depending on the slice, the cell, protocol configuration and the position of the stimulation electrodes, the peak of the EPSCs will *not* be at the same time point in all experiments.

Therefore it is very likely you will have to adjust the positions of the cursors for the EPSC `peaks` now.

Likewise, the positions of the "resistance" cursors may have to be adapted to the protocol settings (timing of the initial depolarization for monitoring the resistance).

After the adjustment, navigate again across the sweeps, to make sure that all cursors are approximately[1] well placed.

Generate epochs

1. 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.
2. 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 7 intervals in each segment.

NOTE: the epochs are *embedded* in the data - hence, they will be saved with your data on disk.

Repeat this step for the `Control/Chase`.

NOTE Because the `Baseline` and `Chase` data *form the same pathway* have the same time base, the cursors will already be visible.

Therefore, **you do not need to set up the cursors again**. However, **you may need to adjust their horizontal coordinates and/or windows**.

Repeat the step again for the `Test/Baseline` and `Test/Chase` data in your `ltp_data`.

NOTE: This time, you will be viewing the "other" pathway - hence with a different time base. Nevertheless, the cursors should have "adapted" their horizontal positions to take into account the new time base. However, *should this not have happened*, then from the signal viewer's menu `Cursors` select `Remove Cursors/Remove all cursors` and set up the cursors again.

Adjust the cursors, then create epochs in `Test/Baseline` and `Test/Chase` as above.

6. Perform LTP analysis

Call

```
result_test, result_control = ltp.LTP_analysis(ltp_data, im_signal, vm_signal, results_basename="result", no
```

where:

`ltp_data:dict` is the averaged block data for control/test pathways (baseline/chase)

`im_signal:int, str` is the index or name of the signal carrying the synaptic responses

`vm_signal:int, str` is the index or [name] of the signal carrying the command voltage (for voltage-clamp experiments). **NOTE**: this is necessary to determine the amount of depolarization used to calculate the resistances as explained here.

`results_basename:str` a meaningful name prefix for the results (default is None)

`normalize:bool` when `True`, the `LTP_analysis` function also normalizes the EPSC amplitudes to the average of EPSC amplitudes over the last 5 minutes of baseline, for each pathway.

The function returns two `DataFrame` objects - one each for the `Test` and the `Control` pathway - with the following numeric columns, containing minute-by-minute values from the baseline reference (-5 minutes before LTP induction) to the end of the chase recordings:

- 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 calculates - all NaNs)
- EPSC0Norm - EPSC0 amplitude normalized to the average EPSC0 over last 5 min of baseline
- EPSC1Norm - EPSC1 amplitude normalized to the average EPSC1 over last 5 min of baseline

These two `DataFrame` obejcts can then be exported as `csv` files directly from Scipyen, then imported into R, etc.

023-04-13 16:05

---

1. Some jittering may occur in the timing of the EPSC peak; this can be compensated for by setting a wider horizontal window for the cursors. To do this, **double-click** on the cursor's vertical line and change the size of the horizontal window. **DO NOT** change the cursor's name. ↩