Roadmap for opto data reformatting to use Plexon Offline Sorter (OFS)

Steps for rosenlab data (TytoLogy):

{summarize here, from document *Processing headfixed tytology data\_8.23.16.docx*}

Development plans for opto data spike sorting using Plexon OFS:

## Step 1: get working for single test of 1 cell from 1 animal

1. export opto *.dat* file to *.nex* file
2. import *.nex* file in OFS
3. export to *.plx* in OFS
4. sort in OFS
5. save sorted data as .*plx*
6. export data as *.mat*
7. incorporate spike times/sorted data/waveforms (?) into *.dat* file.
8. rework analysis program to use sorted data

Step 2: Adapt for use with *all* tests of 1 cell from 1 animal

Uses same general steps as used for single test, with some modifications

1. gather data from all relevant *.dat* and *.mat* files
   1. how to do this?
      1. Select files in GUI? Script? List of files in .txt file?
      2. Rosenlab technique is to copy all files into one directory and then program combines all relevant files
2. export opto *.dat* file to *.nex* file
   1. Rosenlab creates cellinfo.mat file in step 1 and 2 to store information that will be used here
3. import *.nex* file in OFS
4. export to *.plx* in OFS
5. sort in OFS
6. save sorted data as .*plx*
7. export data as *.mat*
8. incorporate spike times/sorted data/waveforms (?) into respective *.dat* file.
9. rework analysis program to use sorted data

Incorporating detected spikes into analysis

Options:

Output file format:

Plx

Mat

Help to have info from export\_for\_plexon() function?

One idea:

Output from OFS (as mat file) is (as noted in Rosenlab AddSpiketimesToMatFiles\_InclUnsort.m)

% 1) Read in .mat files exported by Plexon's Offline Sorter

% Column 1: unit number (where 0 is unsorted)

% Column 2: timestamp where spike crosses threshold (in seconds)

% Columns 3-34 (assuming waveform window of 1311us / 32 samples):

% waveform snippet, with or without prewindow as set in Offline Sorter

% (prewindow default: 494us / 12 samples)

% (window default: 1311us / 32 samples)

% This is in units of samples/sec of raw data file

% (24414.063 Hz based on settings in data acquisition program

% HPSearch or PresentStimCurve in RosenLab)

Note that sampling rate will be different and that timestamp is in seconds

Step 1: knowing each data file’s time “window” (in fileStartTime and fileEndTime), locate the timestamps within this window (column 2). These spikes can then be associated with the appropriate data file.

Step 2: associate each spike down to the appropriate sweep within that file and re-align the spike time to the start of the sweep. Use sweepStartBin and sweepEndBin cells, convert to time in seconds by subtracting 1 and dividing by sample rate

¿how to handle different channels?

10-Feb-20

Need to figure out how to indicate proper channel in the output file from plexon.

e.g., in file `1382\_20191212\_02\_02\_3200.mat’:

output array is named “adc002”

channel number within adc002 array is 3

channels recorded in the MERGED file (imported by plexon OFS) were [4, 5, 11, 14]

and are named spikechan\_4, spikechan\_5, … etc. within the .nex file generated by export\_for\_plexon.m function

10-Mar-20

Working on quick code to get MT data ready for abstract.

NEEDS:

* see waveforms, data (detected spikes, traces) to assess quality
* more individuated control/access to data
* ability to select channel and plot specific raster, psth , curve, FRA, etc etc.

How to do???

testThreshold is turning into overall routine for data specification, loading, thresholding. Consider converting to newer (simpler?) optoproc ish thing

incorporate with SpikeData class???? Probably not easily done, maybe create new class

* should probably create a super class called SpikeData with a skeleton definition (minimal data objects, methods) and rename existing SpikeData to something like PLXData for plexon data, OptoData for opto, etc…. project for LATER!!!

11 March 2020

Creating WAVtestdata class to deal with this.

Some new spike detection/extraction routines downloaded. Need to test …

24 March 2020

Need to map out path forward.

¿spyKing circus – pause for now until using new probes for recording?

If so, need to push on with Plexon export/import and sorting

* will need multichannel, sorted data from MT in order to verify and test
* investigate use of matlab DATASTORE object
  + <https://www.mathworks.com/help/matlab/import_export/what-is-a-datastore.html>

1 April, 2020

* Doesn’t seem to be easy way to have plexon OFS include original A/D channel in the exported .mat file
  + Tried including channel name – no luck, just -1 even when channel name changed to integer from spikechan\_x
  + Emailed plexon for ideas
    - Replies from Mike Barbas at Plexon (6 Apr 2020)

Hello Sharad,

I hope you are doing well during this strange time. This is Mike B. from Plexon. Thank you for your email.

Unfortunately, the “Export Per-“ function in Offline Sorter is unlikely to support the ability to have the desired channel name persist in the “Channel Name” column in the exported Excel file (or Matlab file). This is especially the case for the type of data that you are loading into Offline Sorter, especially since this is a file type that is not commonly tested for the process that you’ve described.

Hello Sharad,

Thanks for your email. Unfortunately, there will not be a way to include the A/D channel number for your specific files when using the “Export Per Waveform” option in Offline Sorter. A peripherally-related fact is that Plexon has an Offline Files SDK, but this SDK is intended for reading PLX and PL2 files specifically acquired via a Plexon data acquisition system.

Best,

**Mike Barbas**

Technical Support Engineer

Company:  +1 (214) 369-4957

Fax:  +1 (214) 369-1775

[mikeb@plexon.com](mailto:mikeb@plexon.com)

7 April 2020

PLX file reading options

Julia module:

https://github.com/simonster/PLX.jl

Patrick Mineault’s blog notes and links:

https://xcorr.net/2011/09/07/reading-plexon-files-in-linux-or-mac/

—also https://xcorr.net/2011/10/13/plexon-matlab-api-now-available-for-linux/

ReallTimeElectrophy:

https://github.com/chrox/RealTimeElectrophy

Benjamin Kraus wrote a MEX function (on Matlab File Exchange):

https://www.mathworks.com/matlabcentral/fileexchange/42160-readplxfilec

FieldTrip project has plx->matlab functionality. windows only?

<http://www.fieldtriptoolbox.org/getting_started/plexon/>

see also http://www.fieldtriptoolbox.org/faq/reading\_is\_slow\_can\_i\_write\_my\_raw\_data\_to\_a\_more\_efficient\_file\_format/

<http://www.fieldtriptoolbox.org/faq/requirements/>

For now, try using B. Kraus readPLXFileC… it seems to work, but need to better understand the options and output format

readPLXFile help:

readPLXFileC - A MEX function to read a PLX file (Plexon, Inc.).

USAGE:

plx = readPLXFileC(filename, varargin)

plx = readPLXFileC('help')

plx = readPLXFileC('version')

INPUT:

filename - Name of the PLX file to read.

varargin - One (or more) of the arguments listed below. Arguments are

parsed in order, with later arguments overriding earlier

arguments.

ARGUMENTS:

'help' - Display this help information

'version' - Display MEX file version information

If 'version' occurs as the first input argument,

the revision number is returned as the first (and only) output,

and the version information is only printed to screen

if no ouptut is requested.

If 'version' occurs after the first input argument,

version information is printed to the screen, but

otherwise the function behaves as though 'version' was not present.

'headers' - Retrieve only headers (default)

(implies 'nospikes','noevents','nocontinuous')

'[no]fullread' - Scan the entire file (default = 'nofullread')

('fullread' is implied if anything other than headers are requested)

'[no]spikes' - Retrieve (or not) spike timestamps (default = 'nospikes')

'nospikes' implies 'nowaves'

'[no]waves' - Retrieve (or not) spike waveforms (default = 'nowaves')

'waves' implies 'spikes'

'[not]units' - Must be followed by a list of units to (not) retrieve

0 = unsorted, 1 = unit 'a', 2 = unit 'b', etc.

'[no]events' - Retrieve (or not) event data (default = 'noevents')

'[no]continuous' - Retrieve (or not) continuous data (default = 'no')

'all' - Read the entire file

(implies 'spikes','waves','events','continuous')

'range' - Time range of data to retrieve

'start' - Start of time range of data to retrieve

'stop' - End of time range of data to retrieve

'first' - First data sample to retrieve

'num' - Number of data samples to retieve

'last' - Last data sample to retrieve

SELECTING CHANNELS:

'spikes','waves','events', and/or 'continuous' can be followed by a

numerical array, which is then parsed to determine which channels to

retrieve. An empty array implies 'no'. If the array is missing,

then all channels are retrieved.

OUTPUT:

plx - A structure containing the PLX file data.

Plx output structure format:

Version: 107

Comment: ''

Date: 7.3788e+05

NumSpikeChannels: 4

NumEventChannels: 4

NumContChannels: 4

ADFrequency: 48828

NumPointsWave: 64

NumPointsPreThr: 24

FastRead: 0

WaveformFreq: 48828

LastTimestamp: 1953119

Trodalness: 1

DataTrodalness: 1

BitsPerSpikeSample: 16

BitsPerContSample: 16

SpikeMaxMagnitudeMV: 3000

ContMaxMagnitudeMV: 5000

SpikePreAmpGain: 1000

AcquiringSoftware: ''

ProcessingSoftware: 'OFS 3.3.5'

SpikeTimestampCounts [n, # channels] List of # of spike timestamps for each channel and unit

SpikeWaveformCounts [n, # channels] List of # of spike timestamps for each channel and unit

EventCounts [# channels] # events per channel … ? not sure

ContSampleCounts [# channels] # of samples for each continuous channel

ContSampleFragments: [1 1 1 1]

**SpikeChannels**: [4x1 struct]

**EventChannels**: [4x1 struct]

ContinuousChannels: [4x1 struct]

FullRead: 1

DataStartLocation: 13952

plx.SpikeChannels(1)

ans =

Name: 'spikechan\_4'

Channel: 1

SIGName: 'adc000'

SIG: 0

SourceID: 2

ChannelID: 0

Comment: ''

NUnits: 3

Ref: 0

Filter: 0

Gain: 0

Threshold: -11508

WFRate: 10

SortMethod: 2

SortBeg: 0

SortWidth: 64

Template: [5x64 double]

Boxes: [5x2x4 double]

Fit: [5x1 double]

Timestamps: [78x1 uint32]

Units: [78x1 uint8]

Waves: [64x78 int16]

p.ContinuousChannels(1)

Name: 'spikechan\_4'

Channel: 0

SpikeChannel: 1

SourceID: 101

ChannelID: 0

Comment: ''

Enabled: 1

ADFrequency: 48828

ADGain: 580

PreAmpGain: 1

Timestamps: 0

Fragments: 1953120

Values: [1953120x1 int16]

Potential solution:

Use

plx = readPLXFileC(<filename>, ‘all’)

to read in everything, including continuous data.

10 Apr 2020:

This seems like it can work! See test\_readPLX.m for example.

Will need to modify the import\_from\_plexon function. Probably best to use a SpikeInfo class method

13 Apr 2020:

Working on stimulus on/off timestamps for export to .nex file

* reorganized/cleaned up export\_for\_plexon after implementing buildChannelData as a method in CurveInfo
* now need to:
  + implement stimulus on/off computation for each sweep , and
  + overload method in WAVInfo in order to account for varied wav stimulus durations

17 Apr 2020:

WAVInfo updated, needs to be tested

<https://www.mathworks.com/support/bugreports/1382373>

22 Apr 2020:

* still getting Matlab crashes – after 2015a “upgrade” using the opto rig license.
  + Might need to use linux laptop?
* SpikeData object now has stimulus onset/offset (stimStartBin, stimEndBin)
  + These are converted to timestamps (in stimStartTimeVector) and saved as event in the exported nex file.
* Now, where were we… oh, importing plx information.

30 Apr 2020:

PLXData object successfully reads in data from .plx files and stores in a struct, P.

Data organization is completely different from the exported mat format file from OFS.

Two options: work with data in P, or “reexport” data in P to the mat format.

Advantages: mat format already has some code to pull out channels, spikes into curves.

Will need to write export/conversion code (not a major deal). But doesn’t seem to have PCA info. Necessary? Could either have separate method for spike data, OR have dummy columns

4 May 2020

Writing mat export method

¿how to scale waveforms? For now, divide by spike chan max value

Also, this file ('1407\_20200309\_03\_01\_1350\_BBN-sorted.ch4,5,7,15.plx') has clipped spikes – double check that plx is ok re gain optimize

11 May, 2020

Clipping was due to older file. Newer files ok

Idea:

More explicit event marker/timestamps to enable more self-contained plx data. ?

Will this work for spyking circus?

Installing spyking circus on petrol laptop \*(Windows)

12 May 2020

Installed anaconda + spyking circus on petrol laptop (linux Debian).

Considering event update, wait to discuss with jeff and

Plexon Machine:

With change in drivers on plexon machine in E117, remote desktop works again. Wtf??????

Need more hard drive space on plexon – the C drive is nearly full. Can we delete datawave stuff from Jazz, Emily, etc???? or install new hd?

Need to create export for spyking…. Similar to plexon, but different….

14 May, 2020

<https://spyking-circus.readthedocs.io/en/latest/code/fileformat.html>

RAW\_BINARY (read/parallel write)

| The parameters for RAW\_BINARY file format are:

|

| -- sampling\_rate -- <type 'float'> [\*\* mandatory \*\*]

| -- data\_dtype -- <type 'str'> [\*\* mandatory \*\*]

| -- nb\_channels -- <type 'int'> [\*\* mandatory \*\*]

|

| -- data\_offset -- <type 'int'> [default is 0]

| -- dtype\_offset -- <type 'str'> [default is auto]

| -- gain -- <type 'int'> [default is 1]

------------------------------------------------------------------

## Raw binary File

The simplest file format is the raw\_binary one. Suppose you have N channels

*𝑐*0,*𝑐*1,...,*𝑐𝑁*

And if you assume that *𝑐𝑖*(*𝑡*)

is the value of channel *𝑐𝑖*

at time t, then your datafile should be a raw file with values

*𝑐*0(0),*𝑐*1(0),...,*𝑐𝑁*(0),*𝑐*0(1),...,*𝑐𝑁*(1),...*𝑐𝑁*(*𝑇*)

This is simply the flatten version of your recordings matrix, with size N x T

Note

The values can be saved in your own format (int16, uint16, int8, float32). You simply need to specify that to the code

As you can see by typing:

>> spyking-circus raw\_binary -i

------------------------- Informations -------------------------

| The parameters for RAW\_BINARY file format are:

|

| -- sampling\_rate -- <type 'float'> [\*\* mandatory \*\*]

| -- data\_dtype -- <type 'str'> [\*\* mandatory \*\*]

| -- nb\_channels -- <type 'int'> [\*\* mandatory \*\*]

|

| -- data\_offset -- <type 'int'> [default is 0]

| -- dtype\_offset -- <type 'str'> [default is auto]

| -- gain -- <type 'int'> [default is 1]

------------------------------------------------------------------

There are some extra and required parameters for the raw\_binary file format. For example, you must specify the sampling rate sampling\_rate, the data\_dtype (int16, float32, …) and also the number of channels nb\_channels. The remaining parameters are optional, i.e. if not provided, default values written there will be used. So the mydata.params file for a mydata.dat raw binary file will have the following params in the [data] section:

file\_format = raw\_binary

sampling\_rate = XXXX

data\_dtype = XXXX # should be int16,uint16,float32,...

nb\_channels = XXXX # as it can not be guessed from the file, it has to be specified

data\_offset = XXXX # Optional, if a header with a fixed size is present

gain = XXXX # Optional, if you want a non unitary gain for the channels

Warning

The raw\_binary file format is the default one used internally by SpyKING CIRCUS when the flag overwrite is set to False. This means several things

* data are saved as float32, so storage can be large
* we can not handle properly t\_start parameters if there are streams in the original data. Times will be continuous
* this is currently the **only** file format properly supported by phy and MATLAB GUIs, if you want to see the raw data

## Configuration File

The code, when launched for the first time, generates a parameter file. The default template used for the parameter files is the one located in /home/user/spyking-circus/config.params. You can edit it in advance if you are always using the same setup.

To know more about what is in the configuration file, [documentation on the configuration](https://spyking-circus.readthedocs.io/en/latest/code/config.html)

# Configuration File

This is the core of the algorithm, so this file has to be filled properly based on your data. Even if all key parameters of the algorithm are listed in the file, only few are likely to be modified by a non-advanced user. The configuration file is divided in several sections. For all those sections, we will review the parameters, and tell you what are the most important ones

## Data

The data section is:

file\_format = # Can be raw\_binary, openephys, hdf5, ... See >> spyking-circus help -i for more info

stream\_mode = None # None by default. Can be multi-files, or anything depending to the file format

mapping = # Mapping of the electrode (see http://spyking-circus.rtfd.org)

suffix = # Suffix to add to generated files

overwrite = True # If you want to filter or remove artefacts on site. Data are duplicated otherwise

output\_dir = # By default, generated data are in the same folder as the data.

parallel\_hdf5 = True # Use the parallel HDF5 feature (if available)

Warning

This is the most important section, that will allow the code to properly load your data. If not properly filled, then results will be wrong. Note that depending on your file\_format, you may need to add here several parameters, such as sampling\_rate, data\_dtype, … They will be requested if they can not be infered from the header of your data structure. To check if data are properly loaded, consider using [the preview mode](https://spyking-circus.readthedocs.io/en/latest/GUI/python.html) before launching the whole algorithm

Parameters that are most likely to be changed:

* file\_format You must select a supported file format (see [What are the supported formats](https://spyking-circus.readthedocs.io/en/latest/code/fileformat.html)) or write your own wrapper (see [Write your own data format](https://spyking-circus.readthedocs.io/en/latest/advanced/datafile.html))
* mapping This is the path to your probe mapping (see [How to design a probe file](https://spyking-circus.readthedocs.io/en/latest/code/probe.html))
* stream\_mode If streams in you data (could be multi-files, or even in the same file) should be processed together (see [Using multi files](https://spyking-circus.readthedocs.io/en/latest/code/multifiles.html))
* overwrite If True, data are overwritten during filtering, assuming the file format has write access. Otherwise, an external raw\_binary file will be created during the filtering step, if any.
* ouput\_dir If you want all the file generated by SpyKING CIRCUS to be in a particular directory, instead of next to the raw data
* parallel\_hdf5 Try to use the option for parallel write of HDF5. Need to be configured (see [how to install hdf5](https://spyking-circus.readthedocs.io/en/latest/introduction/hdf5.html))

## Detection

The detection section is:

radius = auto # Radius [in um] (if auto, read from the prb file)

N\_t = 5 # Width of the templates [in ms]

spike\_thresh = 6 # Threshold for spike detection

peaks = negative # Can be negative (default), positive or both

dead\_channels = # If not empty or specified in the probe, a dictionary {channel\_group : [list\_of\_valid\_ids]}

Parameters that are most likely to be changed:

* N\_t The temporal width of the templates. For in vitro data, 5ms seems a good value. For in vivo data, you should rather use 3 or even 2ms
* radius The spatial width of the templates. By default, this value is read from the probe file. However, if you want to specify a larger or a smaller value [in um], you can do it here
* spike\_thresh The threshold for spike detection. 6-7 are good values
* peaks By default, the code detects only negative peaks, but you can search for positive peaks, or both
* dead\_channels You can exclude dead channels either directly in the probe file, with the channels list, or with this dead\_channels parameter. To do so, you must enter a dictionary of the following form {channel\_group : [list\_of\_valid\_ids]}

## Filtering

The filtering section is:

cut\_off = 300, auto # Min and Max (auto=nyquist) cut off frequencies for the band pass butterworth filter [Hz]

filter = True # If True, then a low-pass filtering is performed

remove\_median = False # If True, median over all channels is substracted to each channels (movement artefacts)

common\_ground = # If you want to use a particular channel as a reference ground: should be a valid channel number

Warning

The code performs the filtering of your data writing on the file itself. Therefore, you must have a copy of your raw data elsewhere. Note that as long as your keeping the parameter files, you can relaunch the code safely: the program will not filter multiple times the data, because of the flag filter\_done at the end of the configuration file.

Parameters that are most likely to be changed:

* cut\_off The default value of 500Hz has been used in various recordings, but you can change it if needed. You can also specify the upper bound of the Butterworth filter
* filter If your data are already filtered by a third program, turn that flag to False
* remove\_median If you have some movement artefacts in your in vivo recording, and want to substract the median activity over all analysed channels from each channel individually
* common\_ground If you want to use a particular channel as a reference, and subtract its activity from all others. Note that the activity on this particular channel will thus be nul

## Triggers

The triggers section is:

trig\_file = # External stimuli to be considered as putative artefacts [in trig units] (see documentation)

trig\_windows = # The time windows of those external stimuli [in trig units]

trig\_unit = ms # The unit in which times are expressed: can be ms or timestep

clean\_artefact = False # If True, external artefacts induced by triggers will be suppressed from data

dead\_file = # Portion of the signals that should be excluded from the analysis [in dead units]

dead\_unit = ms # The unit in which times for dead regions are expressed: can be ms or timestep

ignore\_times = False # If True, any spike in the dead regions will be ignored by the analysis

make\_plots = # Generate sanity plots of the averaged artefacts [Nothing or None if no plots]

Parameters that are most likely to be changed:

* trig\_file The path to the file where your artefact times and labels. See [how to deal with stimulation artefacts](https://spyking-circus.readthedocs.io/en/latest/code/artefacts.html)
* trig\_windows The path to file where your artefact temporal windows. See [how to deal with stimulation artefacts](https://spyking-circus.readthedocs.io/en/latest/code/artefacts.html)
* clean\_artefact If you want to remove any stimulation artefacts, defined in the previous files. See [how to deal with stimulation artefacts](https://spyking-circus.readthedocs.io/en/latest/code/artefacts.html)
* make\_plots The default format to save the plots of the artefacts, one per artefact, showing all channels. You can set it to None if you do not want any
* trig\_unit If you want times/duration in the trig\_file and trig\_windows to be in timestep or ms
* dead\_file The path to the file where the dead portions of the recording, that should be excluded from the analysis, are specified. . See [how to deal with stimulation artefacts](https://spyking-circus.readthedocs.io/en/latest/code/artefacts.html)
* dead\_unit If you want times/duration in the dead\_file to be in timestep or ms
* ignore\_times If you want to remove any dead portions of the recording, defined in dead\_file. See [how to deal with stimulation artefacts](https://spyking-circus.readthedocs.io/en/latest/code/artefacts.html)

## Whitening

The whitening section is:

spatial = True # Perform spatial whitening

max\_elts = 10000 # Max number of events per electrode (should be compatible with nb\_elts)

nb\_elts = 0.8 # Fraction of max\_elts that should be obtained per electrode [0-1]

output\_dim = 5 # Can be in percent of variance explain, or num of dimensions for PCA on waveforms

Parameters that are most likely to be changed:

* output\_dim If you want to save some memory usage, you can reduce the number of features kept to describe a waveform.

## Clustering

The clustering section is:

extraction = median-raw # Can be either median-raw (default), median-pca, mean-pca, mean-raw, or quadratic

sub\_dim = 10 # Number of dimensions to keep for local PCA per electrode

max\_elts = 10000 # Max number of events per electrode (should be compatible with nb\_elts)

nb\_elts = 0.8 # Fraction of max\_elts that should be obtained per electrode [0-1]

nb\_repeats = 3 # Number of passes used for the clustering

make\_plots = # Generate sanity plots of the clustering

merging\_method = nd-bhatta # Method to perform local merges (distance, dip, folding, nd-folding, bhatta)

merging\_param = default # Merging parameter (see docs) (3 if distance, 0.5 if dip, 1e-9 if folding, 2 if bhatta)

sensitivity = 3 # The only parameter to control the cluster. The lower, the more sensitive

cc\_merge = 0.95 # If CC between two templates is higher, they are merged

dispersion = (5, 5) # Min and Max dispersion allowed for amplitudes [in MAD]

smart\_search = True # Parameter to activate the smart search mode

Note

This is the a key section, as bad clustering will implies bad results. However, the code is very robust to parameters changes.

Parameters that are most likely to be changed:

* extraction The method to estimate the templates. Raw methods are slower, but more accurate, as data are read from the files. PCA methods are faster, but less accurate, and may lead to some distorted templates. Quadratic is slower, and should not be used.
* max\_elts The number of elements that every electrode will try to collect, in order to perform the clustering
* nb\_repeats The number of passes performed by the algorithm to refine the density landscape
* smart\_search By default, the code will collect only a subset of spikes, randomly, on all electrodes. However, for long recordings, or if you have low thresholds, you may want to select them in a smarter manner, in order to avoid missing the large ones, under represented. If the smart search is activated, the code will first sample the distribution of amplitudes, on all channels, and then implement a rejection algorithm such that it will try to select spikes in order to make the distribution of amplitudes more uniform.
* cc\_merge After local merging per electrode, this step will make sure that you do not have duplicates in your templates, that may have been spread on several electrodes. All templates with a correlation coefficient higher than that parameter are merged. Remember that the more you merge, the faster is the fit
* merging\_method Several methods can be used to perform greedy local merges on each electrodes. Each of the method has a parameter, defined by merge\_param. This replaces former parameters sim\_same\_elec and dip\_threshold
* dispersion The spread of the amplitudes allowed, for every templates, around the centroid.
* make\_plots By default, the code generates sanity plots of the clustering, one per electrode.

## Fitting

The fitting section is:

amp\_limits = (0.3, 30) # Amplitudes for the templates during spike detection

amp\_auto = True # True if amplitudes are adjusted automatically for every templates

collect\_all = False # If True, one garbage template per electrode is created, to store unfitted spikes

ratio\_thresh = 0.9 # Ratio of the spike\_threshold used while fitting [0-1]. The lower the slower

Parameters that are most likely to be changed:

* collect\_all If you want to also collect all the spike times at which no templates were fitted. This is particularly useful to debug the algorithm, and understand if something is wrong on a given channel
* ratio\_thresh If you want to get more spikes for the low amplitudes templates, you can decrease this value. It will slow down the fitting procedure, but collect more spikes for the templates with

an amplitude close to threshold

## Merging

The merging section is:

erase\_all = True # If False, a prompt will ask you to remerge if merged has already been done

cc\_overlap = 0.85 # Only templates with CC higher than cc\_overlap may be merged

cc\_bin = 2 # Bin size for computing CC [in ms]

default\_lag = 5 # Default length of the period to compute dip in the CC [ms]

auto\_mode = 0.75 # Between 0 (aggressive) and 1 (no merging). If empty, GUI is launched

remove\_noise = False # If True, meta merging will remove obvious noise templates (weak amplitudes)

noise\_limit = 0.75 # Amplitude at which templates are classified as noise

sparsity\_limit = 0.75 # Sparsity level (in percentage) for selecting templates as putative noise (in [0, 1])

time\_rpv = 5 # Time [in ms] to consider for Refraction Period Violations (RPV) (0 to disable)

rpv\_threshold = 0.02 # Percentage of RPV allowed while merging

merge\_drifts = True # Try to automatically merge drifts, i.e. non overlapping spiking neurons

drift\_limit = 0.1 # Distance for drifts. The higher, the more non-overlapping the activities should be

To know more about how those merges are performed and how to use this option, see [Automatic Merging](https://spyking-circus.readthedocs.io/en/latest/code/merging.html). Parameters that are most likely to be changed:

* erase\_all If you want to always erase former merging, and skip the prompt
* auto\_mode If your recording is stationary, you can try to perform a fully automated merging. By setting a positive value, you control the level of merging performed by the software. Values such as 0.75 should be a good start, but see see [Automatic Merging](https://spyking-circus.readthedocs.io/en/latest/code/merging.html) for more details. The lower, the more the merging will be aggressive.
* remove\_noise If you want to automatically get rid of noise templates (very weak ones), just set this value to True.
* noise\_limit normalized amplitude (with respect to the detection threshold) below which templates are considered as noise
* sparsity\_limit To be considered as noisy templates, sparsity level that must be achieved by the templates. Internally, the code sets to 0 channels without any useful information. So the sparsity is the ratio between the number of channels with non-zero values divided by the number of channels that should have had a signal. Usually, noise tends to only be defined on few channels (if not only one)
* time\_rpv When performing merges, the code wil check if the merged unit has a valid ISI without any RPV. If yes, then merge is performed, and otherwise this is avoided. This is the default time using to compute RPV. If you want to disable this feature, set this value to 0.
* rpv\_threshold Percentage of RPV allowed while merging, you can increase it if you want to be less stringent.
* drift\_limit To assess if a unit is drifting or not, we compute distances between the histograms of the spike times, for a given pair of cells, and assess how much do they overlap. For drifting units, they should not overlap by much, and the threshold can be set by this value. The higher, the more histograms should be distinct to be merged.

## Converting

The converting section is:

erase\_all = True # If False, a prompt will ask you to export if export has already been done

sparse\_export = True # If True, data for phy are exported in a sparse format. Need recent version of phy

export\_pcs = prompt # Can be prompt [default] or in none, all, some

export\_all = False # If True, unfitted spikes will be exported as the last Ne templates

Parameters that are most likely to be changed:

* erase\_all If you want to always erase former export, and skip the prompt
* sparse\_export If you have a large number of templates or a very high density probe, you should use the sparse format for phy
* export\_pcs If you already know that you want to have all, some, or no PC and skip the prompt
* export\_all If you used the collect\_all mode in the [fitting] section, you can export unfitted spike times to phy. In this case, the last N templates, if N is the number of electrodes, are the garbage collectors.

22 May 2020

Working on displaying sorted spikes along with continuous data, sweep by sweep

Issue: plexon truncates output data fractional sample rate

e.g., TDT spike channel sample rate is 48828.125 but plexon OFS specifies as 48828.125

need to resample on export!!!

Trying to do this on imported plx file data, but probably best to do on export

9 June 2020

Probably best to deal with resampling of data in read\_data\_for\_export, as this is called by both export scripts (export\_for\_plexon, export\_raw)

Problem: can’t use next lowest integer –

Original Sample Rate: 48828.1250

Resampled SampleRate: 48828.0000

p =

390624

q =

390625

Error: 0.000000000000

Error using upfirdn>validateinput (line 129)

The product of the downsample factor Q and the upsample factor P must be less

than 2^31.

Error in upfirdn (line 81)

[p,q] = validateinput(x,h,varargin);

Error in resample>uniformResample (line 228)

y = upfirdn(x,h,p,q);

Error in resample (line 102)

[varargout{1:max(1,nargout)}] = uniformResample(varargin{:});

Need to recode to have user-specified rate!

* Created functions to speed up (marginally) resampling. Convolution is the main issue – maybe faster computer helps?
* Other question: is plexon really truncating sampling rate or is it just the way the plx file is being read in (i.e., uint16)…??
* Going to try creating a dummy file to test.

sd

Created fake data file: 1407\_20200309\_03\_01\_1350\_TIMETESTDATA.nex

Has 1ms dur spikes at every stimulus onset (100 ms stim delay)

Ran OFS – difference between stim onset and spike time is ~400 us

So: no issues. Carry on….

11 June 2020

Will keep resampling functionality (though default is no resample) for now

Things to do:

* Validate/invalidate sweeps (trials) due to things like motion artifact.
  + Priority: 2
  + Requirements:
    - a graphical way to scroll through reps and identify bad trials
    - plot relevant chunk of continuous data as well as extracted spike waveforms and display spike time/occurance
  + Where to save this information? In SpikeData? CurveData? ….
* Analysis
  + Priority: 1\*\*\*
  + Need to:
    - Select electrode/channel, unit to plot
    - Show waveform overlay in plot
    - How to batch and export?
  + Some issues to resolve:

Since the data, once exported to .nex, are in a pseudo-continuous format, it might be better (and future-proof) to treat them as continuous data. In which case, there are no “trials” or sweeps, and data segments from the recordings are extracted based on a stimulus onset timestamp. Similarly, unit timestamps are then expressed relative to this stimulus onset timestamp.

To facilitate this, timestamps in the nex file need to have sufficient information about the stimulus – identity, amplitude – to be able to pull out desired data segments.

15 June 2020

* Should add file id to Spikes table in SpikeData?
* Some issues:
  + Need dummy data to test things – multi units, multi channel
  + Fix SpikeData problem with specifying (or not) channels and units
  + Data structure: Right now, spikes are in matlab table format. SpikeData provides an interface for accessing the data. Is there a better way?
    - <https://neurodatawithoutborders.github.io/matnwb/tutorials/html/ecephys.html>
    - See also brainstorm-IN  
      <https://www.nature.com/articles/s41597-019-0242-z>

16 June 2020

Generated multichannel, multiunit test data. Sorted data

sortedPath = '/Users/sshanbhag/Work/Data/TestData/working/FakeData/TestData';

rawPath = sortedPath;

nexPath = sortedPath;

nexInfoFile = '1407\_20200309\_03\_01\_1350\_TESTDATA\_nexinfo.mat';

nexFile = '1407\_20200309\_03\_01\_1350\_TESTDATA.nex';

plxFile = '1407\_20200309\_03\_01\_1350\_TESTDATA-Sort.plx';

17 June 2020

getSpikesByStim (presently a script) is trying to do everything. Maybe change it to simply get relevant spikes for a single channel/unit combination This could then be called by a different function or script to get data for all channels,units in this format. Might not be most efficient bt is probably most modular approach.

Data can be returned as a struct… or …. ?

18 June 2020

Working on curves

For WAV data, will need to figure out how to deal with the different levels. optoproc\_plotPSTH\_WAVbyLevel shows one way to deal with this, but should probably be done in CurveInfo and subclasses method getStimulusIndices…

Would it be best to move getSpikesForStim guts into curveData?

Not sure.. branching to work on FRAInfo subclass

22 Jun 2020

* Had to add constructor to FRAInfo
* Reexported, sorted, imported 1407 test data, both FRA only and MERGED all data
* Seems to export, import ok. Now to fix plotting as hoped…

24 June 2020

* Problem: if changes are made to objects, nexinfo.mat files will be outdated, and data need to be reexported to nex. Sorting again shouldn’t be necessary, however
  + Solution: need way to simply remake nexinfo.mat file and not generate .nex file.
* Dealing with spiketimes/spiketable issue: created convertSpikeTableToSpikeTimes method in CurveInfo. To be subclassed as needed. Only FRAInfo should need to do this. For CurveInfo, used code from getSpikesByStim in SpikeData.
  + Inelegant but works for now
  + Works for BBN
  + Works for FRA
  + Probably need a different approach for WAV…….
* Merged into work\_readPLXfile branch. Marked FRAInfo branch for deletion

25 June 2020

Creating demo script to summarize how to plot sorted data

29 June 2020

Finished work on psth, fra, other curves as per optoproc.

Now need to finish putting stimulus timestamps into .nex file

30 June 2020

Algorithm for stim timestamps implemented for bbn, need to duplicate and migrate to subclasses

1 July 2020

Added code for freq, wav. For FRA, there will be 126 event types. Too many???

Add as is for now then test in plexon….

2 July 2020

Had to rethink: SpikeData does not exist in export\_for\_plexon function – only SpikeInfo class is instantiated in export\_for\_plexon.

Instead, moved stimEventTimesForFile moved to SpikeInfo.

Tested on 1407 files – looked at nex file in plexon OFS, seems ok.

Need to tweak names ? or not? Having test as part of event name help distinguish them…

Create FreqTuningInfo class

8 Jul 2020

Plotting implemented in SpikeData.plotAllData method (separate file)

Final thing: implementing saving of plot files. See optoproc for possible code to pull in

13 Jul 2020

repositories to “fetch” (aka update) along with required branch

optosort install

OptoAnalysis working

PlotTools install

UtilitiesToolbox installed

Created spyKingExport branch

For Brainstorm (can use .plx files), need to create events struct array

<https://neuroimage.usc.edu/brainstorm/e-phys/ConvertToBrainstormEvents?highlight=%28events%29%7C%28file%29>

Each entry should have the following fields:

1. **label**: The label of the spike **events**. It should read: “Spikes Channel channelLabel” where channelLabel is the label of the electrode. In the case where multiple neurons are recorded from a single electrode, the label of the **events** should be: “Spikes Channel channelLabel |iNeuron|”, where channelLabel is the label of the electrode and iNeuron the number of the neuron on that electrode.
2. **times**: a ROW vector (1xn where n is the number of spikes) with the timestamps of the spikes that the neuron fired, in seconds.
3. **epochs**: a ROW vector (1xn) with the epoch that each spike occurred (just assign the number 1 to each one and it will work).
4. **color**: A 1x3 vector with the color that the event will be displayed on the Brainstorm viewer. The vector takes values [0,1].
5. **reactTimes**: set this to [].
6. **select**: set this to 1.
7. **channels**: a ROW cell-array (1xn) of empty matrices
8. **notes**: a ROW cell-array (1xn) of empty matrices

The electrode with the label “raw 1” on this example **events** **file**, has 3 neurons that fired 2253, 159 and 83 spikes respectively during this recording. The electrode with label “raw 11” picked up 1181 spikes assigned to a single neuron/cluster etc.

The **file**name of the **events** that needs to be saved in the end, should start with the string "**events**". Example: “**events**\_**file**name.mat” and should contain the **events** structure as described above.

Other stuff: