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