ABITools User Guide Version 0.2

David B. Stockton

July 2, 2017

1 Introduction

This document describes the design and use of the CellSurvey and ABIApiML tools as well as a few other utilities. The NeuroManager investigation database extension is described in a separate Addendum to the NeuroManager User Guide [Stockton, 2017].

2 The ABICellSurvey Database Creation Tool

2.1 Overview

The ABICellSurvey tool creates a local MySQL database of metadata, data, and extracted features using a list of specimen IDs supplied as an input list by the user. The list can be compiled by hand using the ABI CellTypes website, or automatically using RESTful queries [ABI, 2017]. As a simple but useful example, on the page `http://celltypes.brain-map.org/`, one can set filters to choose cells based on broad parameters, then download the data into a comma–separated data file, which then can be imported into Microsoft Excel for examination and transfer to the **CreateCSdb.py** script or equivalent (see below).

The tool connects to the MySQL database server using configuration data supplied the user. If the named database doesn't exist, the tool will create it, then add the tables required for the CellSurvey database.

The ABICellSurvey tool is written in Python and is composed of several files, as follows. The files can easily be edited to add other features of interest to the user. Setting the usePP flag to True in any of the files will print out a lot of ABI structure, helping the user see which features are available. In addition, the ABI SDK source code is available through ABI for examination and possible modification/extension. Additional database functions are easy to create using the supplied ones as a model. Please note that, in general, the ABI Python SDK is compatible only with Anaconda Python (see `http://alleninstitute.github.io/AllenSDK/install.html`), and this is our experience.

As a side effect, the ABI code creates a local "database" of NWB files for each specimen on the user's list. Those files are then available for the ABIApiML package seen in Section 3 as well as direct observation using HDFView.

The database can hold data of any stimulus type but the feature extraction code only works properly (for our purposes, at least) for Long Square types, and so we have only performed feature extractions for those sweeps. Most of the other stimulus types are placed in the database without the feature extractions. The user could easily modify this behavior.

- **CSdbConfig.py** holds the user configuration information for database access separately from the code for security reasons.
- **CreateCSdb.py** is the main file in which the user specifies the ABI manifest file, names the database, and selects which specimens to access.
- **ABICellSurvey.py** defines the *CreateDB* function which performs the primary flow of local database creation: connection to and possible creation of the database, dropping and creation of tables in the database, and population of the database tables from CellTypes experimental data.

- **CellSurveyTableOps.py** is the interface between the *CreateDB* function and the MySQL database. It creates the MySQL queries for all the database operations.
- **ABISweepFX.py** defines two functions. *getABIAnalysisPoints* defines the hardcoded timing associated with each stimulus. *ExtractSweepFeatures* performs sweep–level feature extraction on the supplied waveforms.

2.2 STGFeatExtr.py

We also provide a Python file for use with the NeuroManager *userSimulation()* function. **STGFeatExtr.py** performs ABI feature extraction on remotely–produced simulation raw data files then constructs a JSON–format results file. The Simulator class uploads **STGFeatExtr.py** as part of simulator construction; *userSimulation()* runs the code, then the Simulator class downloads the resulting JSON file. The simulator's *postDownloadProcessingSimulatorSpecific()* method/stage is then used to extract the contents of the JSON file for database insertion and subsequent processing.

2.3 Features extracted

The CellSurvey database holds a subset of the features available, and gathers together features accessed using various of the SDK routines. In many cases, for disambiguation or clarification, we have renamed variables in the Experiment feature extractions. Table 1 lists the CellSurvey variables, their origin, and the original names, together with a description if deemed necessary. The specimen features have not been changed, or are obvious.

2.4 Example Queries

Show all the donors and their fields:

SELECT * FROM donors;

Show all the specimens and their fields:

SELECT * FROM specimens;

Show all the experiments and their fields:

SELECT * FROM experiments;

Show all the experimentFXs and their fields:

SELECT * FROM experimentFXs;

Show all the specimenFXs and their fields:

SELECT * FROM specimenFXs;

Show all experiment feature extractions, prefaced with the associated specimen ID and experiment ID:

SELECT specimens.abiSpecimenID,experiments.abiExpID,experimentFXs.* FROM ((experimentFXs INNER JOIN experiments ON experimentFXs.expIDX=experiments.expIDX) INNER JOIN specimens ON specimens.specIDX=experiments.specIDX);

Table 1: Experiment feature extraction details. The CellSurvey database name, the original ABI name (if any), and a brief description of the feature. Additional information can be seen in the ABI SDK source code and the ABI Electrophysiology white paper [ABI, 2015].

MySQL Name	ABIName	Description | Units
analysisStart	analysis_start	Time from beginning of sweep to start analysis | seconds
analysisDuration	analysis_duration	Time from beginning of analysis to end analysis | seconds
stimulusStart	n/a	Time from beginning of sweep to first nonzero stimulus voltage (not including test pulse) | seconds
adaptation	adaptation	See ABI [2015] | 1/sec
avgFiringRate	rate_avg	Average spike firing rate during analysis window | spikes per second
avgHlfHgtWidth	half_height_width	Average half–height width of all spikes in the analysis window | seconds
baseV	base_v	"baseline voltage, 100ms before stim" – **feature_extractor.py** in ABI SDK | mV
maxSpkV	f_peak	peak voltage of tallest spike detected in the sweep | mV
ISIFirst	first_isi	Interspike interval of first ISI in sweep | msec
ISIMean	mean_isi	Average interspike interval for sweep | msec
ISICV	isi_cv	Coefficient of variation for all ISIs in sweep | unitless
latency	latency	Time delay from start of analysis window to first spike | msec
stimulusLatency	n/a	Time delay from start of stimulus to first spike | msec
frstSpkThresholdV	threshold	Threshold voltage of first spike in sweep | mV
hasSpikes	n/a	was at least one spike detected in this sweep? | boolean
numSpikes	n_spikes	number of spikes detected in this sweep | spikes
hasBursts	n/a	was at least one burst detected in this sweep? | boolean
numBursts	n_bursts	number of bursts detected | burst occurrences
maxBurstiness	max_burstiness_index	max of " 'burst_index' [which] is a comparison index between the highest instantaneous rate within the burst vs the highest instantaneous rate outside the burst." — **ephys_features.py** in ABI SDK | unitless
hasPauses	n/a	was a pause detected in this sweep? | boolean
numPauses	n_pauses	number of pauses detected in this sweep | pauses
pauseFraction	pause_fraction	"average fraction of time spent in a pause" — **ephys_extractor.py** in ABI SDK | unitless
hasDelay	n/a	was a delay detected in this sweep? | boolean
delayRatio	delay_ratio	"ratio of latency to tau (higher means more delay)" — **ephys_extractor.py** in ABI SDK | unitless
delayTau	tau	"dominant time constant of rise before spike" — **ephys_extractor.py** in ABI SDK | seconds

Table 2: Imposed NULL conditions details. In both CellSurvey and the Investigation Databases, for consistency we impose a NULL on a feature under certain logical conditions relating to feature extraction success, existence of spikes, number of spikes, and existence of parent feature. For example, “maxBurstiness” is meaningless if there are no bursts; in that case we put a NULL in the “∄ parent feature” column. In this table, we list the feature MySQL name, then the condition. In general, there is an implied “OR” between columns, however there is some obvious left-to-right increasing specificity as well. Note that this imposition is on top of the ABI result. In other words, if there is no parent feature, we ignore the ABI value for a property of that parent feature (if the code even returns one) and place a NULL in the table. This is accomplished in ExtractSweepFeatures() (in ABISweepFX.py) by setting the feature to None; the connector turns it into a NULL. The reverse is not true; if there is no imposition for a case, the ABI value is passed through whatever it is; if it does not exist, then it is inserted with a NULL. *-also involves (latency > ISIMean) checks.

	Imposed NULL Conditions				
MySQL Name	FX Failure	numSpikes == 0	PrSpk Failure	∄ parent feature	Addl criteria
analysisStart					
analysisDuration					
stimulusStart					
adaptation	✓	✓	✓		numSpikes <3
avgFiringRate	✓		✓		
avgHlfHgtWidth	✓	✓	✓		
baseV	✓				
maxSpkV	✓	✓	✓		
ISIFirst	✓	✓	✓		numSpikes <2
ISIMean	✓	✓	✓		numSpikes <2
ISICV	✓	✓	✓		numSpikes <2
latency	✓	✓	✓		
stimulusLatency	✓	✓	✓		
frstSpkThresholdV	✓	✓	✓		
hasSpikes	✓				
numSpikes	✓				
hasBursts	✓	✓	✓		fx fail
numBursts	✓	✓	✓		fx fail
maxBurstiness	✓	✓	✓	✓	fx fail
hasPauses	✓	✓	✓		fx fail
numPauses	✓	✓	✓		fx fail
pauseFraction	✓	✓	✓	✓	fx fail
hasDelay	✓	✓	✓		
delayRatio	✓	✓	✓	✓	fx fail*
delayTau	✓	✓	✓	✓	fx fail*

Show the Specimen ID, Experiment ID, and number of spikes, for all specimens that have a hero sweep with more than three spikes:

```sql
SELECT specimens.abiSpecimenID, experiments.abiExpID, experimentfxs.numSpikes FROM (((specimens INNER JOIN specimenfxs ON specimens.specIDX=specimenfxs.specIDX) INNER JOIN experiments ON experiments.specIDX=specimens.specIDX) INNER JOIN experimentfxs ON experiments.expIDX=experimentfxs.expIDX) WHERE specimenfxs.hero_sweep_id = experiments.abiExpID AND experimentfxs.numSpikes >3;
```

Show the Specimen ID, Experiment ID, number of spikes, mean ISI, and ISI-CV for all sweeps in specimen 484635029 (that have feature extractions):

```sql
SELECT specimens.abiSpecimenID, experiments.abiExpID, experimentfxs.numSpikes, experimentfxs.ISIMean, experimentfxs.ISICV FROM ((specimens INNER JOIN experiments ON specimens.specIDX = experiments.specIDX) INNER JOIN experimentfxs ON experiments.expIDX = experimentfxs.expIDX) WHERE specimens.abiSpecimenID=484635029;
```

3 ABIApiML package

3.1 Introduction

This document describes the ABIApiML package, which is used to access, using MATLAB only, downloaded NWB-format electrophysiology files from the Allen Brain Institute Cell Types Database. This package does not access the website nor does it download files from the database.

3.2 Basic format of the files

The files are in Neurodata Without Borders (NWB) format, which itself is in the HDF5 format. Top–level groups include stimulus, acquisition, epochs, and specific metadata. Electrophysiology sweeps are organized by number and their data are accessed via different groups. For example, a smoke test sweep called 'Sweep_0' has its stimulus waveform stored under */stimulus/presentation/Sweep_0/data* and the recorded response stored under */acquisition/timeseries/Sweep_0/data*.

Experiments are located under */epochs*. Like Sweep_0, Experiment_15, say, will have its its stimulus waveform stored under */stimulus/presentation/Sweep_15/data* and the recorded response stored under */acquisition/timeseries/Sweep_15/data*, but in addition the consumer has access under the experiment itself. So the stimulus is also located at */epochs/Experiment_15/stimulus/timeseries/data*, and the response is also located at */epochs/Experiment_15/response/timeseries/data*.

Analysis in the form of spike detection for sweeps that have spikes is located under */analysis/aibs_spike_times/Sweep_X*.

We see that the sweep number associates a stimulus waveform+metadata, acquisition-response waveform+metadata, and analysis spike time list with one another. Sweep numbers are not necessarily contiguous, although every stimulus sweep that is present has a corresponding acquisition-response sweep. Analysis sweeps only exist for existing experiments that have a response sweep that shows spikes.

All the data uses a standard sampling rate of 200 kHz according to the ABI documentation [ABI, 2015], but this value is not contained within the NWB file.

Many entries in the files contain the message "please see `http://celltypes.brain-map.org/documentation`"; however this address is not valid. The page `http://help.brain-map.org/display/celltypes/Documentation` is currently valid.

3.3 Overview of this package

The ABIApiML package minimizes the need for the MATLAB user to deal with the HDF5 structure of the chosen NWB file. It allows direct access to sweeps by sweep number and to experiments by experiment number. The user's basic approach is to construct an object of the **APICellData** class that specifies the file in question, then use object methods to access the information in the file. The package also uses **ABIExperiment** and **ABISweep** classes to access the data in the file. This package does not write to the NWB file, but only reads from it. There is an existing Python/MATLAB API/SDK for writing NWB files, but not reading from them [NeurodataWithoutBorders, Teeters, 2017].

There is a system of id numbers in use in the ABI Cell Types database. The electrophysiology page for the chosen cell shows the 'specimen id'. For example, the webpage `http://celltypes.brain-map.org/mouse/experiment/electrophysiology/324466858` shows the electrophysiology page for the slice with specimen id '324466858'. This page has a download link which downloads an NWB–format file called '324466856_ephys.nwb', which the careful reader will notice is not the same number; this is called the 'ephys–result–id'[1]. This package (ABIApiML) works with the NWB file path itself, so whether you have the NWB from download as just explained, or whether it is downloaded via the CellSurvey software as described above (in which case it will be stored as "ephys.nwb" under "celltypes/specimen_SpecimenID/", the approach is the same. All the same, the specimen id is stored in the NWB file, and we have provided two ABICellData class methods for the user to access it — one method returns the number itself (GetSpecimenInfo()), and

[1]This seems to be generally true but may not be true for all entries in the database.

the other opens the ephys webpage for that specimen using the MATLAB settings for default browser (OpenSpecimenWebPage()). In addition, the GetIdentifier() and GetCollectionInfo() methods return the ephys-result–id.

Note that the current version of this package does not provide access to all data within the NWB file. We encourage the user to use a viewer such as HDFView to access the file internals should there be any questions about the information provided or not provided by this package. The NWB file format is documented in NeurodataWithoutBorders [2016]. Not all elements in the NWB files downloaded from ABI are filled–out or fully or consistently (or even exist), and so there may be file–file inconsistencies to deal with. Only by spending a lot of time with the files have we reached this point; if we have misunderstandings or misinterpretations please point them out to us!

3.4 Downloading the NWB file — Summary

The ABIApiML package makes use of a previously downloaded NWB file. The user has several ways to find and download this file, as described here. At the time of this writing, we have not supplied a MATLAB utility to grab an NWB file directly from ABI.

Manually: Access the electrophysiology page for the chosen cell. As described above, this page shows the 'specimen id'. For example, the webpage `http://celltypes.brain-map.org/mouse/experiment/electrophysiology/324466858` shows the electrophysiology page for the slice with specimen id '324466858'. This page has a "Download data" link which downloads an NWB–format file called '324466856_ephys.nwb'. Although the file number is not the same as the speciment id, the file itself will hold the correct specimen id.

Using the ABI RESTful API: Formulate a RESTful query as shown at `http://help.brain-map.org/display/celltypes/API#API-NWB` and present it using cURL or other utility.

Using the Python SDK: The CellTypesCache class of the Python SDK will download the NWB file of a selected specimen. Optionally the class will create a cache, and check that that cache to see if the selected specimen's file has already been downloaded. The **ABICellSurvey.py** file shows an example of this usage. In addition, there is ABI–supplied tutorial code at `http://alleninstitute.github.io/AllenSDK/cell_types.html`.

Using the ABICellSurvey tool: The ABICellSurvey tool makes use of the CellTypesCache class of the Python SDK to maintain a local cache of NWB files. As a result, creating a CellSurvey database for a specific list of specimens will automatically download and cache the NWB files for those specimens. The user can then access those files directly using the ABIApiML package.

3.5 Example workflow outline/highlights

Here is an example of obtaining the NWB file and then accessing it using MATLAB.

3.5.1 Example workflow

1. Go to `http:\celltypes.brain-map.org`, which is the web interface to the Allen Brain Initiative Cell Types database.

2. Pick a cell for your project using the tools available and click on it. This brings the ephys block associated with that cell to the top of the column on the right hand side of the page. Click on that block to go to the electrophysiology summary for that specimen.

3. Click on 'download data' to download the NWB file, which contains the electrophysiology data associated with that specimen. Note: there are other ways to download NWB files; see Section 3.4 in this Guide.

4. Bring up MATLAB and build a script similar to **ExampleScriptABICellData.m**.

5. Run the script to make use of the data found in the NWB file in MATLAB.

3.5.2 Highlights

Among the things you can do within a MATLAB script using ABIApiML are:

- Construct an instance of the **ABISweep** class, if desired, to access directly any single sweep, including the test sweeps and experiment sweeps.
  - Use methods in the **ABISweep** class such as GetBasicInfo() or GetStimulusData() to access data associated with the sweep.
  - Use the GetTimeBase() method to construct a time base for the sweep that is suitable for plotting and other analysis.
- Construct an instance of the **ABICellData** class to gain access to all experiment–associated data in the NWB file. In this context an experiment is basically a single stimulus–response pair, along with metadata.
  - Use the GetExperimentList() method to get a list of all the experiments in the file.
  - Use the GetExpReport() method to get a list of all experiments in the file together with the short description of the stimulus waveform used in each experiment.
  - Use the GetAnalysisSweepList() method to get a list of the sweeps that have detected spikes.
  - Use the GetExperiment() method to construct an **ABIExperiment** object and access the information associated with a specific experiment.
- Construct an **ABIExperiment** object to access the specific sweeps and metadata associated with the experiment.
  - Use methods such as GetExperimentDescription() to access metadata associated with the experiment.
  - Use the GetExperimentSweep() method of the **ABIExperiment** class to get an **ABISweep** object that is specifically associated with that experiment. This object has access to both the stimulus and analysis/acquisition data associated with the experiment.

3.6 Summary of the example script

The file 'ExampleScriptABICellData.m' is an example of basic usage of the ABIApiML API. This file creates an instance of the ABICellData class using a downloaded NWB file, then uses that instance to access the session metadata, the acquistion and stimulus sweeps, and then the experiments of the file. Finally, the script plots the stimulus and response data from an experiment and labels the plot using metadata contained within the NWB file. The script also opens the specimen's ABI CellTypes webpage for comparison.

The plot is formed with a layout that is intended to mirror the layout of the plots seen on the ABI website ephys data for the cell. That is, going to http://celltypes.brain-map.org/mouse/experiment/electrophysiology/324256803 shows one the ephys data webpage for the sample with id of 324256803. Clicking the 'Download data' link on that page downloads the NWB file called '324256801.nwb'. Running the script using that number (324256801) as the ephysResultID, and using 'experiment = 37' in line 53, results in a plot with 36 spikes. Clicking the purple blob next to 'Sweep select' on the ephys data webpage shows the same data in the same format as plotted by the script. In addition, the script prints the value of the 'spiketimes' variable, which lists those times.

Better is the format used by the CellTypesCache class of the SDK, which places "ephys.nwb" files under a directory named with the Specimen ID. We have redone the example script (below) using that setup; the ABICellData constructor handles both by using the absolute path of the NWB file so that it supports both approaches.

The ABICellData class uses **curl** to access the ABI services (to get certain data parameters), and, for simplicity, we have put the curl bin directory as a hardcoded value in the ABICellData properties.

3.7 List of user–accessible methods by class

3.7.1 ABICellData class

- ABICellData(path, ephysresultid) — constructor. Constructs an object of the ABICellData class using the path to the NWB file and the EphysResultID
- GetIdentifier() — returns the top–level identifier string from the file, including the data source (ABI), the date of the data (I believe), and the EphysResultID, which is also the filename
- GetNWBVersion() — the version of the NWB used by this file
- GetSpecimenInfo() — returns general info about the specimen, including specimenID
- OpenSpecimenWebPage() — opens the ABI webpage for the specimen represented by this NWB file and this instance of the ABICellData class
- GetCollectionInfo() — returns general info about this data collection, including start time, sessionID, and protocol
- GetSubjectData() — returns general info about the subject from whom this specimen was taken, including species, genotype, age, and sex
- GetExperimentList() — returns a list of the experiments represented in this file
- GetExperimentReport() — returns an array associating each experiment number with its stimulus description
- GetExperiment(expnum) — returns an ABIExperiment object for the indicated experiment number
- IsExperiment(expnum) — a boolean indicating whether or not the given expnum is actually an experiment in the file
- GetAcquisitionSweepList() — returns a list of sweeps in the acquisition group
- GetStimulusSweepList() — returns a list of sweeps in the stimulus group
- GetAnalysisSweepList() — returns a list of sweeps in the analysis group
- GetAcquisitionSweep(sweepnum) — returns an ABISweep object for the indicated sweep number in the acquisition group
- GetStimulusSweep(sweepnum) — returns an ABISweep object for the indicated sweep number in the stimulus group
- GetAnalysisSweep(sweepnum) — returns an ABISweep object for the indicated sweep number in the analysis group
- IsAcquisitionSweep(sweepnum) — a boolean indicating whether or not the given sweepnum is actually a sweep in the acquisition group
- IsStimulusSweep(sweepnum) — a boolean indicating whether or not the given sweepnum is actually a sweep in the stimulus group
- IsAnalysisSweep(sweepnum) — a boolean indicating whether or not the given sweepnum is actually a sweep in the analysis group

3.7.2 ABIExperiment class

- ABIExperiment(filepath, expnum) — constructor. Constructs an ABIExperiment object for the given experiment number using the NWB file found at filepath. The user can use this constructor directly or, as seen in the example file 'ExampleScriptABICellData.m', make use of the ABICellData method GetExperiment(expnum)
- GetExpNum() — returns the experiment number for an object of the ABIExperiment class
- GetExpStr() — returns the string used in the NWB file for the specific experiment embodied by an object of the ABIExperiment class
- GetExperimentDescription() — returns the description of the experiment as contained in the NWB file
- GetExperimentTimes() — returns the start and stop times of the experiment
- GetTimeBaseWindow() — for a given sweep contained in an experiment, the actual experiment waveform is a central cutout of the entire sweep's data. This method returns the start and stop indices of this cutout
- GetExperimentSweep(expnum) — returns an ABISweep object for the sweep associated with this experiment object
- GetStimulusDescription() — returns the AIBS Stimulus information for the experiment's sweep
- visualize() — plots the experiment stimulus and response in a two-subplot format (see Figure 1)

3.7.3 ABISweep class

- ABISweep(filepath, sweepnum, fromexperiment, expnum) — constructor. Constructs an object of the ABISweep class for the given sweep number using the NWB file found at filepath. fromexperiment is a boolean that, if true, indicates that the sweep is associated with an experiment and that the experiment group pathways in the NWB file should be used
- GetSweepNum() — returns the sweep number associated with an object of the ABISweep class
- GetSweepStr() — returns the string used in the NWB file for the specific sweep embodied by an object of the ABIExperiment class
- GetSamplingRate() — returns the sampling rate for the sweep. For the ABI Cell Types database, this number is always 200000 Hz.
- GetSamplingPeriod() — returns the reciprocal of the sampling rate
- GetBasicInfo() — returns basic information about the sweep, including gain, starting_time, and num_samples
- GetAIBSStimulusInfo() — returns the AIBS information associated with the sweep, such as description, interval, and name
- GetCapacitances() — returns the capacitances associated with the sweep
- GetResistances() — returns the resistances associated with the sweep
- GetElectronics() — returns the bias_current, bridge_balance, and capacitance_compensation of the sweep

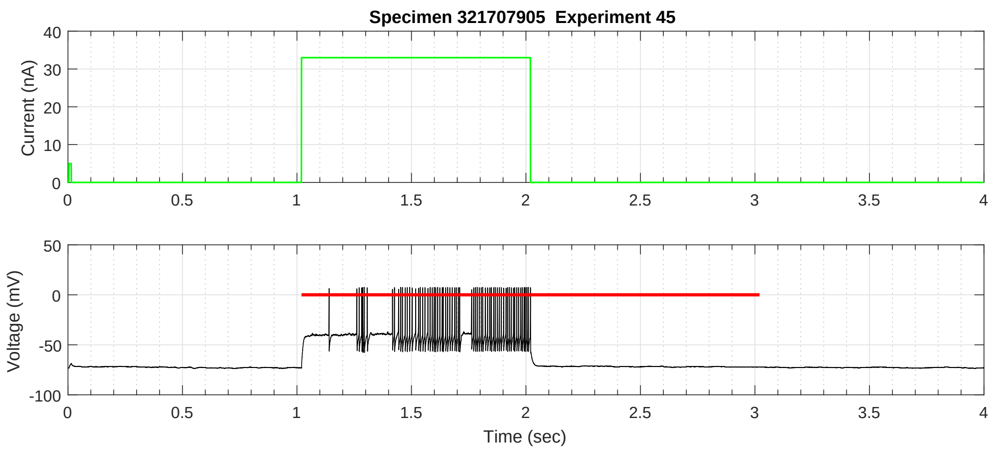

Figure 1: Experiment class visualize() method example. The visualize() method is a quick way to see an experiment's stimulus and response waveforms from downloaded NWB files, and is part of our example of use of the ABIApiML class files. The green waveform is the current stimulus waveform known as "Long Square", the black waveform is the response voltage recording, and the red line marks the analysis window position with respect to both. Please see Section 3.7.2 class.

- GetTimeBase(tfUseStartTime) — returns a MATLAB array of length num_samples (see GetBasicInfo() above) with the sampling time for each datapoint (as calculated using the starting_time and the sampling period). If tfUseStartTime is false, will use 0.0 as the starting point instead, which is compatible with the spike times provided in the analysis group (see GetAnalysisSpikeTimes() below)
- GetStimulusData() — returns an array with the data from the stimulus data associated with this sweep
- GetAcquisitionData() — returns a MATLAB array of length num_samples with the data from the acquisition/response data associated with this sweep
- GetAnalysisSpikeTimes() — returns an array with the analysis–detected spike times associated with this sweep

4 Common sweep feature extraction

We use common code for sweep feature extraction from experimental raw data and simulation results raw data. We supply with this package the Python function called "ExtractSweepFeatures"; it calls the ABI Python SDK directly. The NeuroManager Addendum associated with the ABITools project [Stockton, 2017] describes the simulation side of this. Figure 2 illustrates the two different uses of the ABI SDK through ExtractSweepFeatures().

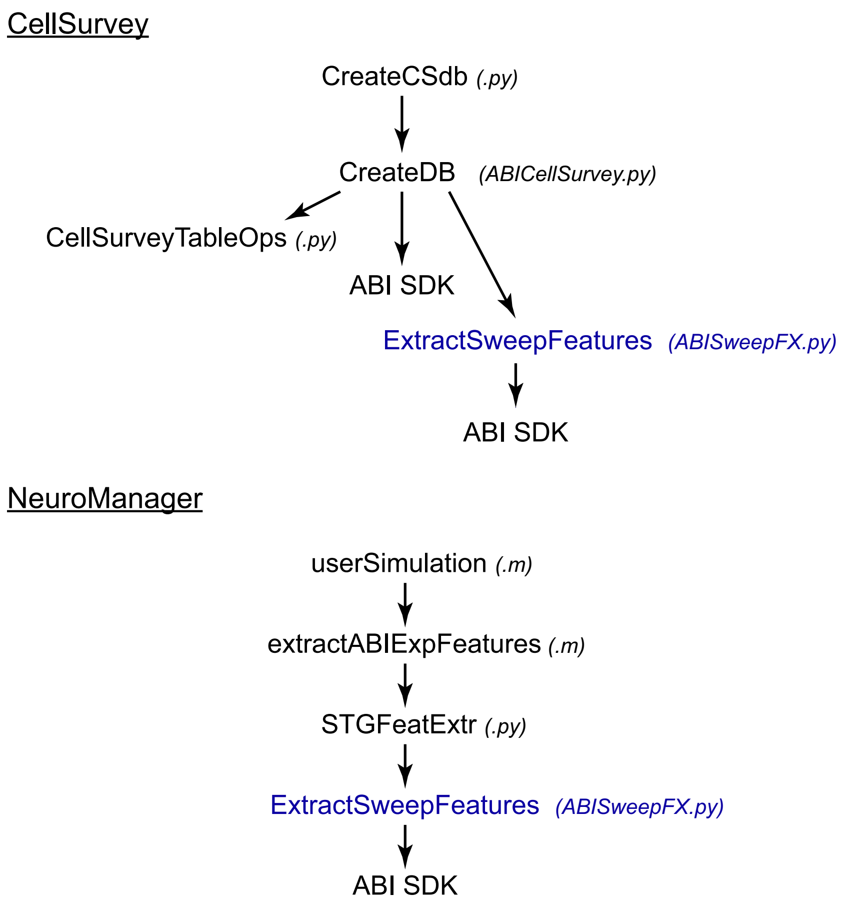
CellSurvey
CreateCSdb (.py)
CreateDB (ABICellSurvey.py)
CellSurveyTableOps (.py)
ABI SDK
ExtractSweepFeatures (ABISweepFX.py)
ABI SDK
NeuroManager
userSimulation (.m)
extractABIExpFeatures (.m)
STGFeatExtr (.py)
ExtractSweepFeatures (ABISweepFX.py)
ABI SDK

Figure 2: The two uses of the ABI SDK Sweep Feature Extraction code. ExtractSweepFeatures() (shown in blue) calls the SDK directly to extract the sweep features listed in Table 1 from either experimental data (top) or simulation raw results (bottom).

5 ABI Electrophysiology Sweep Timing

We spent a lot of time poring through the hdf5 files to determine some basic information about sweep construction. As a natural result, we developed this diagram (Figure 3) which illustrates the timing involved in the ABI sweeps, and offer it as a service to the reader.

Each sweep's stimulus involves a small test pulse followed by the selected stimulus. The stimuli are described in the ABI Ephys White Paper [ABI, 2015]. The SS, LS, and 2s Supra stimuli start at 1.02 seconds from the start of sweep, the 0.5ms Supra at 1.022 seconds, the ramps at 1.0225 seconds, and the triple SS at 2.02 seconds.

The figure also shows that an "experiment" is a trimmed sweep, and what most of those boundaries are.

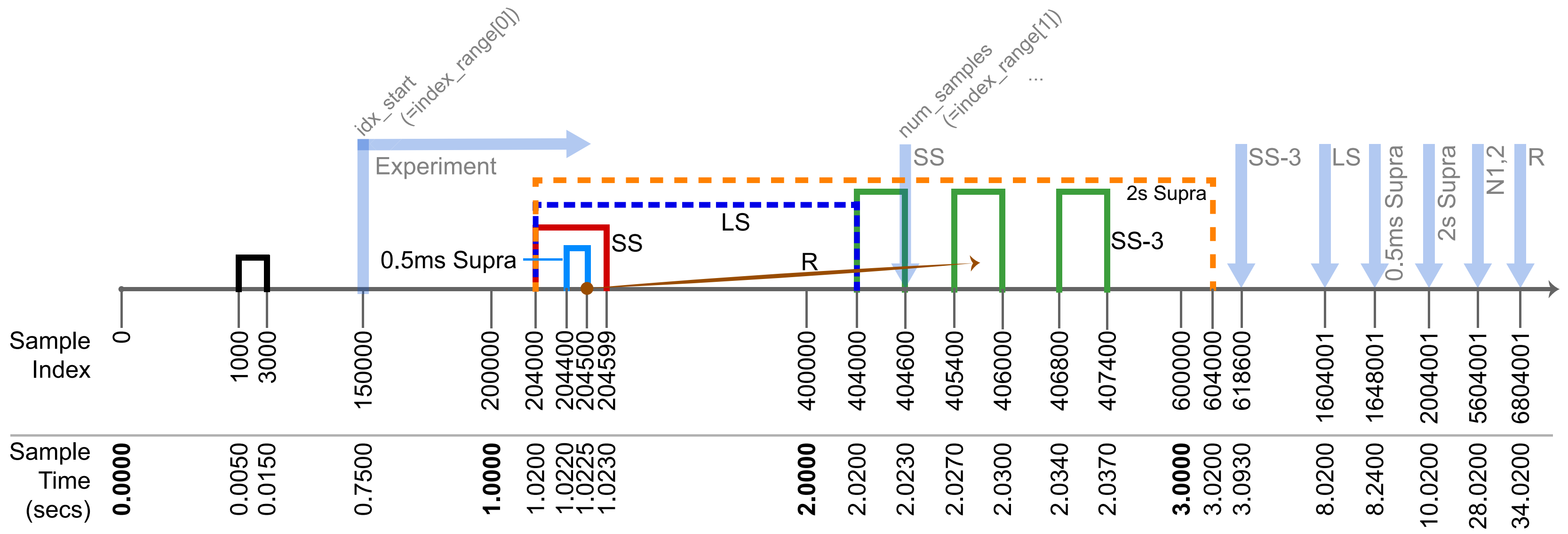

Figure 3: Sweep and experiment timing. The essential sweep and experiment timing as obtained directly from the [specimen].nwb files. The heavy gray line illustrates the timeline associated with a sweep, with sample index and associated time (sampling rate = 200000 Hz) below. Above the heavy gray line in rich colors are the stimulus waveforms: solid black for the short test pulse, solid light blue for "Square - 0.5ms Suprathreshold", solid red for the "Short Square", solid green for the "Short Square - Triple", solid brown for the two Ramp stimuli, dashed blue for the "Long Square", and dashed orange for the "Square - 2s Suprathreshold". Noise 1 and 2 stimuli are not shown. The ABI experiment is a trimmed version of the sweep; its endpoints are stimulus–dependent as illustrated by the transparent wide light blue arrows. For example, Long Square experiments start at sweep index 150000 and end at index 1604001. The comments above the experiment arrows indicate the actual name in the NWB file (idx_start and num_samples), and the values returned by the Python SDK (index_range[0] and index_range[1]). SS: Short Square, LS: Long Square, R: Ramp, SS–3: Short Square Triple, 0.5ms Supra: Square 0.5ms Suprathreshold, 2s Supra: Square 2s Suprathreshold, N1,2: Noise 1 and 2. Height of waveforms depicted are for illustrating timing only and have no association with the actual currents involved. The timeline is not linear; refer to the time and index values below the heavy gray line for actual timing information. For graphical simplicity, some adjacent indexes were combined into a single line. For precise index and time information, please consult the actual NWB files or the accompanying table.

References

ABI. Allen Cell Types Database - Electrophysiology. Technical report, Allen Brain Institute, 2015. URL http://help.brain-map.org/download/attachments/8323525/EphysOverview.pdf?version=1&modificationDate=1456188786670.

ABI. Allen Brain Institute RESTful Model Access (RMA), 2017. URL http://help.brain-map.org/pages/viewpage.action?pageId=5308449.

NeurodataWithoutBorders. Python (and Matlab) API for Neurodata Without Borders (NWB) format - Code. URL https://github.com/NeurodataWithoutBorders/api-python.

NeurodataWithoutBorders. NWB File Format Specification Version 1.0.4beta, 2016. URL https://github.com/NeurodataWithoutBorders/specification/blob/master/version_1.0.4_beta/nwb_file_format_specification_1.0.4_beta.pdf.

David B. Stockton. NeuroManager User and Programmer's Guide – 0.990 Addendum ("ABITools" Upgrade). 2017.

Jeff Teeters. *Python (and Matlab) API for Neurodata Without Borders (NWB) format - Documentation*, 2017. URL http://neurodatawithoutborders.github.io/api-python/build/html/api_usage.html.