Welcome to Pyclamp! This is an pre-Alpha (development) version, and note that presently this manual is incomplete and under revision. Pyclamp is Python package used to extract and analyse electrophysiological data. A graphical user interface has been developed to allow a user to run the package without requiring any knowledge of Python code. Presently, Pyclamp is designed to perform very specific forms of analysis on evoked synaptic responses:

Waveform analysis : This is a highly user-interactive environment that can be used to discriminate synaptic events, obtain various measures of their kinetics and size, and output the results. The most pertitinent measurements exported are MWD (Mean Windowed Discursion) values that can be used for quantal analysis.

Quantal analysis : This part of the package performs both simple variance-mean analysis and Bayesian quantal analysis to estimate the quantal size and number of release sites amongs other measures. It can be used in isolation or in conjuction with the waveform analysis described above.

When Pyclamp is started, you can access the 'File menu' to choose to open to a data (i.e. ADC waveform) or analysis (*i.e.* quantal analysis file); for convenience you can also open a 'results file', which can be used to reload a saved quantal analysis without having re-run the number-crunching. While the quantal analysis component can read the output files generated the waveform analysis, it can also read other tabular and spreadsheet file types. Consequently, the use of the waveform analysis component is not essential to perform quantal analysis.

1 Installation

Presently, Pyclamp is not compiled and must therefore be run from source by Python 2 with all dependencies installed:

- python 2.7.x.
- NumPy 1.8 or later.
- SciPy 0.13.1 or later.
- xlrd 0.9.2 or later.
- PyQt 4.x.
- PyQtGraph 0.9.8 or later and its python-opengl dependency.

Pyclamp supports only 64-bit environments. Since there is no official 64-bit version of NumPy for Windows, Pyclamp most easily runs within Linux. If you do not use Linux, you are welcome to try to Pyclamp on your host operating system but it may not work. There are unofficial 64-bit versions of Numpy for

Windows which can try to make work with Pyclamp. If this is all too much for Windows users, then I might be tempted a compile a stand-alone executable for Windows if enough interest is out there, so please email if you'd like me to do this. If you're Apple user, then life is probably easier for you installing Pyclamp but I'm not in much of a position to help because I've never touched an Apple.

Since there are many different flavours of Linux, it is difficult to provide a comprehensive and accurate description of how the dependencies of Pyclamp should be installed. Many popular Linux distributions come with package managers that will allow you to install the dependencies safely from trusted repositories. For various reasons, it is possible your distribution does not include all the dependencies, particularly PyQt, PyQtGraph, and most elusive of all, the python-opengl dependency. If web-searches for these packages are not sufficient for you to be able to install them, then your choice of Linux distribution might be a bit tricky to accommodate. In such cases, you should consult your local Linux guru.

I haven't yet site-packaged Pyclamp as a setup.py-installable Python package, so installation is nothing more than copying the unzipped three directories 'quantal', 'gen', and 'pyclamp' into a destination directory of your choice. The graphical user interface necessitates Pyclamp to be run within X and not from a virtual console. The workng directory for running Pyclamp has to '.../pyclamp/'. Pyclamp can be started by navigating to the 'pyclamp' directory and running (within Linux) the 'pyclamp' file. Or you can run it from command line in one of three ways:

Method 1:

```
sh-4.2$ cd pyclamp/
sh-4.2$ sh pyclamp.sh
```

Method 2:

```
sh-4.2$ cd pyclamp/
sh-4.2$ python2 pyclampmain.py
```

Method 3:

```
sh-4.2$ cd pyclamp/
sh-4.2$ python2
Python 2.7.5 (default, Dec 1 2013, 00:00:00) [GCC] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> execfile("pyclampmain.py")
```

If all methods fail, you probably have a missing or broken dependency. In such cases, an informative error message will hopefully guide you as to the cause.

2 Waveform data analysis

2.1 Importing data

For waveform analysis, Pyclamp can read gap-free and episodic old- and new-ABF (Axon) files. There's also experimental support for CED's SMR waveform and wavemark data, SES's WinEDR/WinWCP formats, and raw binary (SMRX-readability might be accommodated in future). You may or may not notice more than one file listed in the listbox at the top of the dialogue box. This is an experimental feature that enables contatentation of waveform files that appear to have potentially identical protocols.

Presently, Pyclamp requires one pair of channels to be selected: one passive and one active. These are specified by the user in the 'Channel Selection' part of the 'File and channel selection' dialogue box. The passive channel should contain the recorded waveform data for the evoked synaptic responses. The active channel may comprise of monitored applied voltage/current data, or the activity of a pre-synaptic cell in the case of a pair recording. Note that the user may specify the same channel for the passive and active fields.

If you open a single channel file, the active and passive channel must be the same. If the waveform file contains more than two channels, then you are strongly advised to turn additional passive/active pairs off using the drop-down menus, unless your computer has vast RAM and graphics RAM. The 'Channel Information' part of the dialogue box lists for each the channel the sampling interval, gain, and offset. Note that the values can be changed at this stage in case they are incorrect. Following this dialogue box is another in which you are prompted to specify the clamp and configuration of the passive channel.

2.2 Waveform navigation

Waveform navigation is performed by mouse movements, mouse clicks, and mouse wheel movements. It can take some getting used to. However with practice, users will quickly become more efficient in waveform navigation in comparison to navigating through traditional interfaces based on icons and scrollbars.

Pyclamp benefits substantially from the interactivity of Luke Campagnola's PyQtGraph, which is a Python package that exploits the functionality of all three mouse buttons. This may make things tricky for users of Apple mice. Clicking while keeping the mouse motionless performs different functions depending on the button:

Left-click: Select episode.

Middle-click: Toggle between concatenated and overlayed display.

Right-click: Brings up the context menu to 'View All'.

The other options in the context menu, seen after clicking the right mouse button, are not recommended. Note that the instead of using the middle button, a user may press the spacebar on the keyboard to toggle between concatenated and overlayed displays. Note that the waveform data is colour-coded according whether an episode occurs at the beginning (red) or end (blue) of a recording. While this is self-evident in the concatenated view, the colour code can be helpful in the overlayed view to detect gradual changes over time.

The mouse-wheel performs different navigation functions depending on the direction:

Wheel-up: Zoom in.

Wheel-down: Zoom out.

If a mouse is moved while a button is depressed, or 'dragged', the specific navigation performed depends on the button:

Left-drag: Windowed-zoom.

Middle-drag: Pan.

Right-drag: x (vertical) and y (horizontal) zoom/unzoom.

The effect of windowed-zoom performed by the left-drag is dependent on the direction:

Down and right (or while pressing both 'Meta' and 'Alt'): Zoom in both x and y directions.

Up and right (or while pressing 'Meta'): Zoom only in x direction.

Down and left (or while pressing 'Alt'): Zoom only in y direction.

Up and left: View all.

Windows users may want to know that the 'Meta' key is in fact what is commonly called the 'Windows' key. Note that the 'Meta' and 'Alt' keys can be used to during all mouse movements to lock the y and x axes respectively. The only exception to this is if the left button is clicked with the mouse motionless, which performs a 'view all' navigation. It's also possible to 'select' single events using the left mouse button. Multiple events can be selected by holding 'Shift' (or toggled using 'Control') before using left mouse button to drag a frame around the events you wish to select. There are also hotkeys for selection:

'U' key: select all episodes.

'I' key: invert current selection.

'O' key: unselect all episodes.

Finally, two keyboard buttons can be used to remove/return selected waveform episodes:

'Delete' key: delete selected episode.

'Insert' key: return most recently deleted episode.

Apart from these keys, none of the operations described above actually change the waveform data. There a number of buttons towards the bottom screen, that allow the user to manipulate the waveform data, but note that these changes in no way affect the original wave data file.

2.3 Restore

This button restores the waveform data to the original file contents. A partial restore is also possible if you specify at which point of the log you wish to revert to.

2.4 Trim

For large files with many long episodes, it is often inefficient to deal with entire traces when the excerpt of interest occurs over a very short period. After clicking the 'Trim' button, you have to select which channel you will use to specify the period of interest. Note that this waveform data manipulation will affect not only the selected channel but all channels.

2.5 Filter

Within the precision of the waveform datatype, Pyclamp can apply a low-pass or high-pass filter to the waveform data for any channel. After clicking this button, your must specify the channel from the drop-down menu and the remainder dialogue box allows customisation of the filter. You must specify the bandpass type and enter the desired corner (or 'cutoff') frequency in Hertz. Pyclamp uses a digital filter that includes a transition band alongside the corner frequency within the stopband. The width of the transition band can be set (relative to the corner frequency) and the non-linearity of its magnitude bode pole profile can be adjusted (0% is fully linear). Since the entire channel is filtered, it is usually a good idea to trim the data before filtering. If you try to filter the entire contents of a huge data file, do not be surprised if Pyclamp takes a long time or freezes your machine.

2.6 Trigger

A single y-value can be used to serve as an oscilloscope-style trigger. The dialogue box prompts you to specify the relevant channel and any quiescent duration or 'dead time', which enforces a minimum time between triggers within each episode. Note that you can trigger on descents as well as ascents, or alternatively use 2- or 3-point discrimination to trigger on differences rather than absollute values. Following this dialogue box, a cursor will appear for the selected channel for you to set the trigger points; for 2- or 3-point triggers, you will need to specify 3 or 4 points respectively. Trigger marks themselves are marked as capital 'T's above the waveform trace of the relevant channel (you may need to zoom-out to see them). The trigger marks are not in themselves particularly useful unless you align by triggers (see below), or export the trigger times.

2.7 Align

Pyclamp can re-align episodes according the time-point of the maximum or minimum waveform value of a selected channel over a specific time window. You must specify the relevant channel in drop-down menu of the dialogue box and selected whether you wish to align by the maximum or minimun value. If triggers marks are present, you may also align on the basis of triggers. The length of excerpts of trace kept before and after each point of alignment can be specified from the 'Limit per-alignment window' drop-down. Pyclamp also allows you to set triggers by the point of alignment within each episode. Following the dialogue box, a vertical cursor will appear for the selected channel and you must indicate the region for maximum/minimum/trigger detection using two left mouse clicks. Note that all other channels will be aligned accordingly. If you align by triggers, the number of resulting episodes will correspond to the number of triggers in the specified region; this will eliminate episodes of the original data with no triggers in the specified region or give rise to overlapping episodes in cases in which there is more than one trigger in the specified region. Note that all alignment procedures automatically detect coincidental (fully overlapping) episodes and eliminate duplicates. This can be useful for removing multiple-triggering events.

2.8 Baseline

The baseline subtraction function allows you to offset individual episodes so that the mean value over a specified window is zero. Consequently, it only affects the data of one channel, which must be selected in the dialogue box. Following this dialogue box, a vertical cursor will appear and you must indicate the region for baseline subtraction using two left mouse clicks. There are other 'experimental' baselining options, such as using two windows or subtracting fits to single or double exponentials fitted over specified regions; look carefully at the results to make sure that the program have achieved what you intended.

2.9 Selection

The above functions are provided (with other more experimental ones) in the 'Process' menu. The 'Selection' menu is useful for selecting individual events that can be marked for 'deletion' to remove unwanted events or set to 'active' or 'inactive', for example if you to flag events as being 'successes' as distinct from 'failure' when it comes to outputting tabulated results. More sophisticated methods of selection are provided by a subapplication that can discriminate in

2 or 6 dimesions. The subapplication is still very much in its infancy. While it is far from completed, you can make use of some of its functionality. The discriminator is a descendent of similar programs designed to isolate action potentials called SpikeLab and LabSpike (for details see Bhumbra, Inyushkin, and Dyball (2004) J. Neuroendo. 16:390-397)

From the dialogue box, you have to select the channel you wish to use for discrimination. Using two left mouse clicks to position vertical cursors, you must specify the region of the episodes to be used for the purposes of discrimination. Pyclamp will then calculate a number of parameters that characterise the waveform shape in the specified window of each episode for the selected channel. Depending on the amount of data, these parameters may take some time to evaluate.

After the shape parameters are calculated, you will be presented with either one or three scatter plots on two waveform plots. In the case of a single plot, you simply discriminate using the ellipse with each episode represented as a dot. Each ellipsesand be moved by its edges or reshaped by dragging one of the two anchors on their major or minor axes. By manouvring the ellipses, you can 'cluster-cut' signal from noise. Right-clicking an ellipse restores it to its original position.

For 6D discrimination, dots are coloured-coded according to whether they lie inside or outside the three ellipses. The inside/outside colouring can be reversed by clicking the middle-mouse button for the relevant display. Dots are coloured logically according to the Boolean logic for all three ellipses:

Red : Red only.

Green : Green only.

Blue : Blue only.

Magenta : Red and blue only.

Yellow : Red and green only.

Cyan : Green and blue only.

White : Red, green, and blue.

Grey : None.

Note that individual dots can selected and the corresponding trace is shown in the bottom left panel. Below this panel are buttons that indicate the numbers of dots correspond to each colour. All traces corresponding to a given colour can be overlayed by clicking on the button of the appropriate colour in the button panel below. Note that the episodes can be deleted or undeleted by pressing the 'delete' or 'insert' button respectively.

The panel in the bottom right overlays 'accepted' traces according whether a given colour is 'On' or 'Off'. You can toggle the accept-state of a colour by clicking on the button of the appropriate colour in the button panel below. Note that in addition to the accepted traces, the selected traces shown in left panel are also overlayed on the right except in a highlighted colour. Finally, the total number of accepted traces is shown on the bottom right button and you can click here to complete the discrimination. Note that any episodes deleted in the discrimination window are deleted irreversibly once you leave this screen. The only way the deleted episodes can be returned to the data set is by clicking on 'Restore', but this will also lose any discrimination you have performed.

2.10 Analysis

In addition to highly 'experimental' analyses, this menu provides measurement techniques of analysis that tabulates results as tab-delimited files, which can be used for subsequently for quantal analysis, namely 'Post-synaptic response' and 'MWD'. Selecting 'Post-synaptic response' will open a dialogue box, where you must selected the channel you wish to analyse. For measurement of the mean window discursion (MWD), you are prompted to choose the polarity and time period. The MWD window is the window over which the wave data is averaged to measure the size of the response for the purposes of quantal analysis. Although it can be decided automatically by Pyclamp by leaving the value 0, you are strongly advised to set this an appropriate value for quantal analysis since the time-period should be constant for different sets of data obtained from a same recording. An optional comment field is also included in this dialogue box, which can serve as a label for the quantal analysis part of the program.

Following this dialogue box, you will have to set the position of two vertical cursors to specify the window for the entire synaptic response. Since kinetic measures include exponential fits, you are strongly advised to include a period of baseline activity just before and just after the period of synaptic response. It may take some time for calculation of the various measures of synaptic responses. Finally you will be presented with a summary figure from which you can open a dialogue box to save the results as tab-delimited text file. Note that if you select a pre-existing file, the file is not overwritten but the results are appended. If you wish to perform quantal analysis for different sets of data obtained from the same recording, you must export all analysis results to the same file.

A spreadsheet program, such as Libreoffice Calc or Microsoft Excel, can be used independently to open the results file. While importing the file, you should specify that 'tab' is the only delimiter. The first few rows will contain the log that steps taken to perform the analysis. This will be followed by a results table with the following headings:

Active : either 'True' or 'False' usually depending on whether the event was 'accepted' during discrimination.

Index : the episode index.

Dis : the waveform value a the the most substantial deflection, whether peak or trough.

Distime : the time in seconds from the beginning of the window of the most substantial deflection.

Max : the maximum waveform value.

Min : the minimum waveform value.

IPR(Rise) : the inter-percentile (20-80) range in the rise period.

IPR(Fall) : the inter-percentile (20-80) range in the fall period.

IPT(Rise) : the inter-percentile (20-80) time in the rise period.

IPT(Fall) : the inter-percentile (20-80) time in the fall period.

IPST(Rise) : the inter-percentile (20-80) start time in the rise period relative to Distime.

IPST(Fall) : the inter-percentile (20-80) stop time in the fall period relative to Distime.

Asymptote(Rise) : the latest asymptotic (baseline) value in the rise period.

Asymptote(Fall) : the earliest asymptotic (baseline) value in the fall period.

Asymptime(Rise) : the latest asymptotic (baseline) time in the rise period relative to Distime.

Asymptime(Fall) : the earliest asymptotic (baseline) time in the fall period relative to Distime.

FitDur : the duration of the dual exponential fit.

FitAmp : the deflection amplitude estimated from the dual exponential fit

FitOffset : the offset of dual exponential fit.

FitLAC : the log amplitude coefficient of the dual exponential fit.

FitLDC(Pos) : the log decay constant of the positive term of the dual exponential fit.

FitLDC(Neg) : the log decay constant of the negative term of the dual exponential fit.

FitLDC(Rise) : the log decay constant for the rise period of the dual exponential fit.

FitLDC(Fall) : the log decay constant for the fall period of the dual exponential fit.

FitMeanAD : the mean absolute deviation for the dual exponential fit.

FitMaxAD : the maximum absolute deviation for the dual exponential fit.

MD : the mean deflection of the entire response throughout the selected period.

MWD : the mean windowed deflection around the maximal response for the selected period.

Note that 'deleted' events are excluded from the results table. If a series of results were appended, then the file will alternate between the log and respective results table. If you input the file for quantal analysis, Pyclamp will use the data under MWD (mean windowed deflection). It is also important to note that quantal analysis requires inclusion of both successes and failures.

If you wish to specify the exact central location for the MWD window, you can click 'MWD' under the 'Analysis' menu instead of performing a full analysis of synaptic responses.

3 Quantal analysis

3.1 Importing data

For quantal analysis, Pyclamp can read the TDF files exported by Pyclamp's 'Post-synaptic response' or 'MWD' analyses. In addition the quantal analysis program can import Microsoft Excel files and tab-delimited files. For Excel files, data must be tabulated in columns so that each column contains measurements obtained from the same condition of release probability. Note that the number of measurements for each condition do not have be the same, and the top row can be used for the purposes for labelling each of the conditions. Tab-delimited files exported from the waveform analysis application are detected specifically and parsed to extract the data for quantal analysis. Raw tab delimited files are by default read as column-major; if this fails a row-major import is attempted.

Pyclamp is not a suitable application to perform multiple-probability fluctuation analysis (MPFA) only, because the estimates for the quantal parameters are not corrected for quantal variances. Uniquely, Pyclamp offers an implementation of Bayesian quantal analysis (BQA) assuming either homogeneous or heterogeneous release probabilities across release sites (for details see Bhumbra and Beato (2013). J. Neurophy. 109:603-320).

3.2 Enter baseline noise

The dialogue box prompts you to enter the standard deviation of the baseline noise. This is necessary for both multiple probability fluctuation analysis (MPFA) and Bayesian quantal analysis (BQA). For MPFA, it fixes the intercept for the parabolic fit, whereas for BQA it is used to fix the variance for the normal distribution centred at zero. The value should be known at the time of analysis. Neither method of quantal analysis provides an estimate for the baseline standard deviation.

The presence of failures however does allow empirical evaluation using one tail for the distribution. The initial value in the dialogue box is based on this assumption, but this value should not be relied on especially if there are few no failures.

3.3 Variance mean plot

The variance-mean plots the two moments of the data sets with variance-errors overlayed as error bars. A quadratic fit is also shown. Coefficients for the quadratic fit is used for multiple-probability fluctuation analysis (MPFA) to estimate the quantal size ($\hat{Q}$) and number of release sites ($\hat{N}$). The estimates are shown to the side of the figure alongside the baseline standard deviation (ϵ) used to fix the intercept. Note these estimates in no way correct for quantal variances. If the estimate for the number release sites is negative, then your data is not suitable for MPFA. If you wish to perform Bayesian quantal analysis, click on the button labelled 'Run quantal analysis'. If you wish to view the amplitude histograms for the different conditions, click 'Histograms'.

3.4 Running quantal analysis

Before running Bayesian quantal analysis (BQA), you are prompted to choose the settings for the calculations using a dialogue box. The 'Data selection' allows deselection and reselection of data sets used for quantal analysis. Note that a minimum of two data sets are required. The remainder of the dialogue box allows the user the customise the nature and precision of the analysis. It makes mathematical sense to make the probability of release to be at a resolution higher than any of the other parameters. In general, 128 is enough. The resolution for the coefficient of variation and certainly the number of release sites should be less or equal to the resolution of the probability of release.

Most critical is specification of the maximum number of release sites. Of course this is dependent on the relevant biological system, but an upper bound should be known. If the value specified is too small, then BQA will be forced into underestimating the number of release sites. An inappropriate high value however comes at the cost of reducing precision of calculation and computational time. In practice, it is best to enter a realistic maximum. Due to the discrete nature of the number of release sites, the ideal maximum of number of release sites fulfils the condition: any integer $* (N - 1) + 1$, where N is the resolution sampling the number of release sites. By default the program uses Jeffrey's rule to assume a reciprocal prior for the number of release sites (you'll need to read about this if you're a bit fuzzy on what this means) but the program gives you the option of assuming a flat prior by unchecking this radiobutton.

For all parameters, a greater resolution improves precision but at the cost of computational time and memory. The heterogeneity sampling resolution defaults to a value 1, which indicates to Pyclamp that the BQA model specification assumes identical release probabilities across all release sites. If you enter a greater integer value, the BQA algorithm will assume heterogeneous release probabilies according to a beta model and sample the parameter α by the specified resolution.

If you model heterogeneous release probabilities beware of two possible issues. The first is simply a matter of sampling resolution. An additional continuous parameter adds another dimension to an already multi-dimensional matrix. Consequently, you should try to select a low sampling resolution (*e.g.* 12) for α and consider dropping the sampling resolution for the coefficient of variation to a similar figure. The second issue is the exponential matrix growth evaluating heterogeneous release probabilities as the maximum number of release sites increases. Since the minimum size of the matrix is 2^n, where n is the number of release sites, the time taken also exponentially grows. While this is trivial for low values of n, the cost of the numerical operations can become substantial once n reaches the high teens.

3.5 Quantal analysis results

When the analysis as completed, you will be presented with the same variance-mean graph with the BQA-projected relation overlayed in green. The results alongside tabulate the results, and in case of BQA results are based on the medians of probability distributions.

N : the range of numbers of observations.

epsilon : the baseline noise standard deviation.

MPFA-Q : the MPFA estimate for the quantal size.

MPFA-N : the MPFA estimate for the number of release sites.

Resn : the resolutions of the four parameters.

r : the BQA estimate for the maximal response.

q : the BQA estimate for quantal size.

gamma : the BQA estimate for the gamma shaping parameter.

alpha : the BQA estimate for the homogeneity parameter.

lambda : the BQA estimate for the gamma scaling parameter.

v : the BQA estimate for the coefficient of variation.

n : the BQA estimate for the number of release sites.

s : the BQA estimates for the probability of release for the different conditions.

As before, the histograms can be displayed. But now they are overlayed by the BQA-projected distributions and below each, the marginal probability distribution for teh probability of release is plotted. The single-dimension marginal distributions can be plotted by selecting ‘Marginal 1D’. Similarly all two-dimensional conditional and joint probability mass distributions can be plotted by selecting ‘Marginal 2D’. In each case, the values for the probability distributions can be exported as a tabulated tab-dellimited text-file by clicking ‘Export’. Finally, the entire BQA results can be saved as a python-pickle file by clicking ‘Save’. Note these files can be loaded from Pyclamp’s main ‘File’ menu by select ‘Open Results File’ without having to perform the entire analysis again.

For those interested, it’s very easy to batch-analyse a directory full of analysis files in a single run and save the results, but you need to know Python. If you’re interested please email me and I’ll show you an example.

4 Disclaimer

This is development version of an experimental program written by a neuroscience academic, and therefore somewhat susceptible to coding bugs. Consequently, the author could not accept any responsibility for any bad things that might result from using it. But in effort to be helpful rather than legally defensive, I would be grateful for any suggestions or bug reports.