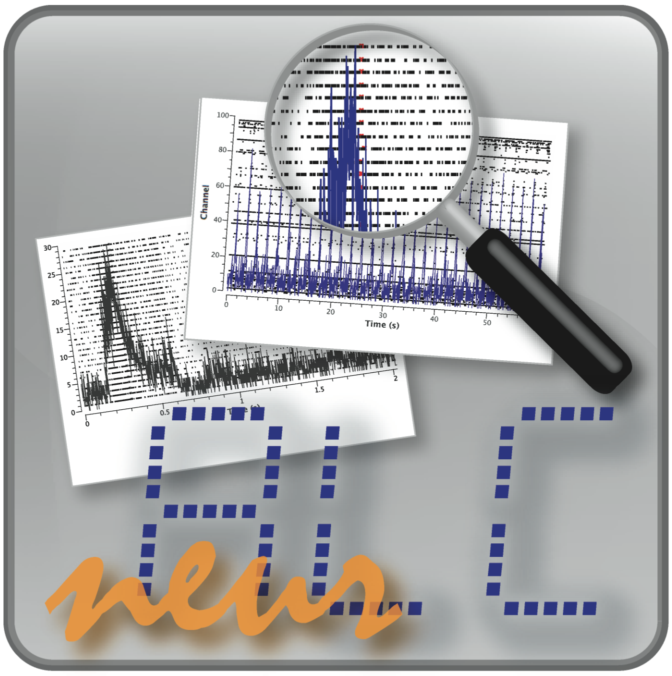


an open-source software for the analysis of multi-electrode recordings

revision of August 22, 2006

Some Small Print

The program is based Trolltech's Qt 3.3 C++ framework[3] and the libraries QWT[4], as well as QWT3D[5] for the graphical user interface. Several implemented functions of the program are taken from the TISEAN package[6]. Additionally parts of the software are based on NEV2lkit[7]. The calculation of fourier transformations uses routines from the FFTW3 library[8]. The unsupervised cluster analysis of the PSTH data is based on Klustakwik[9]. Database functionality is provided by utilizing MySQL[10].

Acknowledgement

*neur*ALC Version 1.0 was developed during july - october 2005 in the group of Prof. Dr. E. Fernandez at the institute of bioengineering of the university Miguel Hernandez, Alicante (Spain). We like to thank everyone who contributed to the making of *neur*ALC.
.

Conventions Within This Manual

- References to components of the GUI, commands or other functions of *neur*ALC are set using a `typewriter` face.
- Aspects we like to emphasize are set using a roman typeface and marked by a marginal exclamation icon (like the one shown on the left).
- The layout of the manual for *neur*ALC is intended for one-sided printing.
- Textual description of menus or command structures uses →-symbols to illustrate a hierarchical implementation.

[1] http://www.gnu.org
[2] http://creativecommons.org/licenses/by-nc-sa/2.0/
[3] http://www.trolltech.com
[4] http://qwt.sourceforge.net
[5] http://qwt3dplot.sourceforge.net
[6] http://www.mpiks-dresden.mpg.de/∼tisean
[7] http://nev2lkit.sourceforge.net/∼NEV2lkit
[8] http://www.fftw.org
[9] http://klustakwik.sourceforge.net
[10] http://www.mysql.com

1 Preface

*neur*ALC - is a program intended for working and analyzing neuronal multi-electrode recordings. In the current version it offers some functionality to reveal and analyze some of the information contained in such recordings and it is further intended as a kind of “catalyst” for the development of a free available cross-platform program for the analysis of electrophysiological recordings, which is distributed under the GPL (GNU Public License 2.0).
This manual is intended as short description of the provided functionality and its technical background, as well as offering a some tutorial and programming information.

The Name...

Well, the program, or better its name come a long way. For a small group of people the software is also known as the program formerly called “WAND”, “PROBE”, or “Genie”. These names were dismissed after we discovered that for example the *Wide Area Network Daemon* was simply there first, and well, there are some trademarks on editable lubricants, etc. which uses some of the names we thought of in the beginning - in short: ***neur*ALC** is composed from two syllables - ***neur*** derived from the consideration that the program works with neuronal data, and **ALC**, which is the abbreviation used in international aviation for the Alicante airport.

2 Installation

*neur*ALC uses Trolltech’s Qt 3.3 C++ framework and the libraries QWT and QWT3D for the graphical user interface. Several implemented functions of the program are taken from the TISEAN package. The calculation of fourier transformations uses routines from the FFTW3 library. Database functionality is provided by utilizing MySQL. In general terms the program should therefore run on any platform on which these packages and libraries are available.
As of this writing *neur*ALC is provided in four different "flavors": binary installers for Linux, MacOSX 10.3 (or higher), Windows 95 (or higher) and a source distribution (if you manage to compile a binary on any other platform, please contact us. We’re happy to integrate and distribute more installers on the *neur*ALC website). Download the package of your choice from `http://neuralc.sourceforge.net`.

2.1 Linux

The binary installer for Linux on x86 processor comes as a gnu-zipped tar archive. After download from sourceforge and dearchiving `neuralc.tgz`, run the `install-neuralc.sh` shell script from within `bash`. Prerequisites for using this binary are an installation of Trolltech’s Qt framework version 3.x (KDE version 3 comes with libraries), the QWT version 0.4.2, QWT3D version 0.2.6 and the FFTW library package. To use the database functionality an additional installation of MySQL (version 3 or higher) is needed. Depending on the Linux-distribution you use, different ways of installing MySQL on your system exist. Refer to `http://www.mysql.com` for additional information.

2.2 MacOSX

The MacOSX installer requires MacOSX 10.3 or higher. The binary installer package includes the program and all required libraries (Qt 3.3.4 incl. support for MySQL, qwt 0.42, qwt3d 0.25, fftw3) for MacOSX . Download the file `neuralc1.0-installer.zip`, dearchive and double click the icon to install. To use the database functionality an additional installation of MySQL (version 3 or higher) is needed. Visit `http://www.mysql.com` and download the MySQL-installer package for MacOSX of your choice.

2.3 Windows

The binary installer for Microsoft Windows is based on NSIS[11] and installs the program, example files, manual, sources and all required libraries. Download `neurALC-win32.exe` from sourceforge, unzip it and click on the installer icon. The program is installed in its own directory in C:\Programs\neurALC and registered as entry within the Windows START-button menu. All required libraries are installed in the windows system directory. To use the database functionality an additional installation of MySQL (version 3 or higher) is needed. Visit `http://www.mysql.com` and download the MySQL-installer package for Windows of your choice.

2.4 Compilation

The program is written in C++. To compile under your flavor of MacOSX, *nix or Windows, first download and install the required libraries QWT, QWT3D and FFTW, as well as MySQL. Then download the *neur*ALC source code-archive. Extract all files to a single directory, open a shell, change to the directory and type `qmake`, then `make`. That should be all you need to do. Under MacOSX and *nix `gcc` is used; to use the provided `.pro`-file under Windows, an installation of the `mingw`-package with such functionality, is recommended. If you run at this stage in to an error, generate a project file using Trolltech's `qmake` or change the supplied `neuralc.pro` according to your Qt installation and C++ compiler requirements. As of this writing the non-commercial version of the Qt 3.2 framework for Windows is available only as a CD-ROM add-on of the book Blanchette, Summerfield: "C++ GUI programming with Qt3", ISBN 0131240722 (which offers in general a very nice introduction to Qt programming). To download the QPL/GPL version for other platforms visit the Trolltech web page.

Please be aware that Qt3.x is neither binary- nor source code-compatible to Qt4. To compile the source code of *neur*ALC, or use one of the provided binaries, an installation of Trolltech's Qt3 is implicitly needed.

[11] http://nsis.sourceforge.net

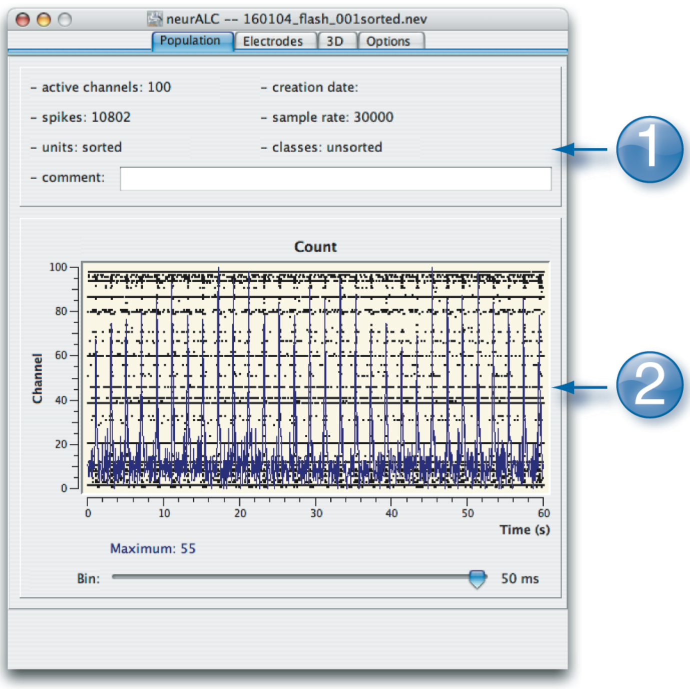
3 Usage&Functionality

3.1 The *neur*ALC GUI

Double click the program icon (shown on the right) to run *neur*ALC. After the program start, open a data file with a neuronal recording in NEV-format (at the moment *neur*ALC can only open recording files in the Neuronal Event Format Version 2). Your screen should look similar to the one shown in the figure 1 (all screen shots in this manual were taken under MacOSX; Trolltech's Qt provides native look&feel for all supported platforms - depending on the operating system you use, the graphical user experience might slightly differ). *neur*ALC's user interface is designed in five (at least we think) logical sections. The main graphical user interface represents the hub to all available visualizations and program parameter. It provides access via four tabs:

3.1.1 The Population-Tab

As default after start-up, *neur*ALC's Population-tab becomes selected/active. It is subdivided in an information (figure 1←1) and a visualization (figure 1←2) part. Information regarding the number of electrodes (`active channels`), date (`creation date`), number of recorded neuronal spikes (`spikes`), `sample rate` (in Hz), and the sorting status of units- (`units:sorted/unsorted`) or TRC (`classes:sorted/unsorted`) for the opened experimental file is provided (as far as it is filed within the file). Additionally an existing


- active channels: 100
- creation date:
- spikes: 10802
- sample rate: 30000
- units: sorted
- classes: unsorted
- comment:

Maximum: 55
Bin: 50 ms

Figure 1: *neur*ALC GUI: Population-tab

comment with up to 256 characters is displayed and can be edited or created via the provided text entry field. The lower part of the tab (figure 1←2) is used to visualize the current data set or analysis result - on the level population or experiments. That implies that the result of any selected calculation or visualization will use all data - the recorded "population' - 'provided by the opened experimental data file. The slider widget at the button of the window allows, if applicable, to interactively adjust specific visualization or analysis parameter.

3.1.2 The Electrodes-Tab

Selecting the Electrodes-tab (Figure 2) allows the user to access, analyze and visualize the information provided in the actual opened experimental file on electrode-, (if previously sorted and comprised in the NEV-file) neuronal unit- , and , if this analysis was performed, temporal response class (TRC)-level. Like for the Population-tab (p.5),

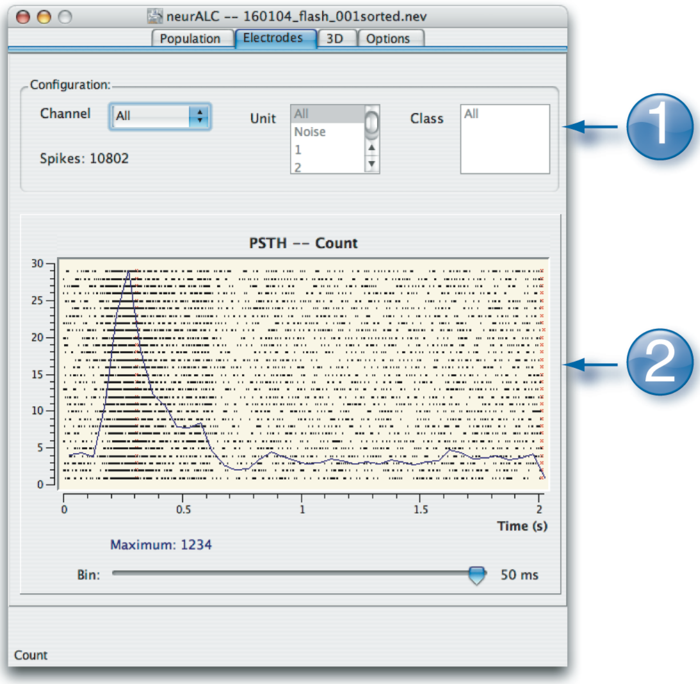


Figure 2: *neur*ALC GUI: Electrodes-tab

the graphical user interface is subdivided in 2 parts. One (figure 2←1) which provides access to the single electrodes (`Channel`), neuronal units (`Unit`; only accessible if this information is contained in the actual opened experimental file), and temporal response class information (`Class`; only accessible if previously calculated and assigned). The lower, second part of the Electrodes-tab serves as display for the visualization of the data or analysis results (figure 2←2) part. The slider widget at the button of the window allows, if applicable, to interactively adjust specific visualization or analysis parameter.

3.1.3 The 3D-Tab

*neur*ALC offers a few visualization in three dimensions. The `Option`-pop-up menu (figure 3←1) in the upper part allows to select one of the provided visualization. Clicking the `Home`-button resets the view of the actual 3D-visualization (figure 3←2) in the lower part of the window. Using the mouse it is possible to rotate, zoom, or navigate oth-

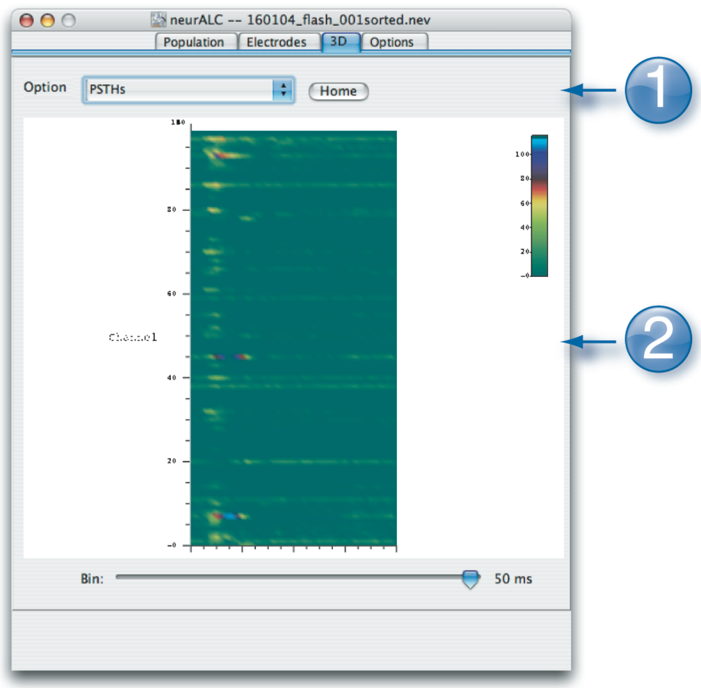


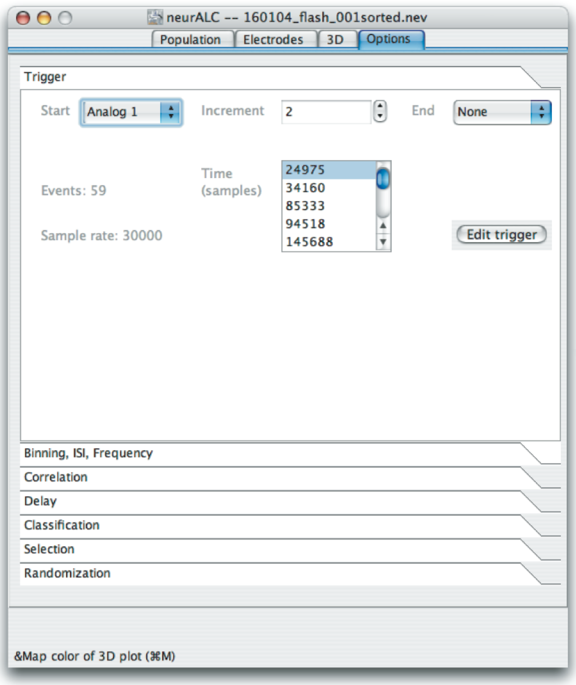
Figure 3: *neur*ALC GUI: 3D-tab

erwise within the three-dimensional visualization. Appendix-7.1 includes a table with *neur*ALC's keyboard short cuts and a description of the possible mouse-based navigation. The slider widget at the button of the window allows, if applicable, to interactively adjust specific visualization or analysis parameter.

Whether in *neur*ALC's Population-, Electrodes- or 3D-tab: any visualization displayed can be saved - depending on the configuration on the system which is used - into a file using the Print-dialogue. Via this OS-provided functionality it is for instance under MacOSX possible to save any plot to a vector-based PDF-file.

3.1.4 The Options-Tab

This tab provides access to all analysis and visualization parameter. The adjustable parameter are arranged in 7 different, tabbed views related to: the `Trigger` information within the opened experimental data file, `Binning - ISI (inter spike interval) - Frequency`, `Correlation`, `Delay`, `Classification`, `Selection`, and `‘‘el Ultimo’’`. Any tuning in one of the options is applied to all related, further calculations or visualization during the actual program session.



Start: Analog 1
Increment: 2
End: None
Events: 59
Sample rate: 30000
Time (samples): 24975, 34160, 85333, 94518, 145688

Figure 4: *neur*ALC GUI: Options-Tab

Trigger This tab allows to select and adjust the stimulus related time code (“trigger” information) which is filed within the actual opened experimental data file (or used for the “automatic” analysis of a directory of files). The NEV2-format definition[12] allows to store 2 different data types with this information: “analog” and “digital” data packets. Up to 5 differently tagged analog (`Analog 1`, `Analog 2`, etc.), and one digital packet identifier (tagged as `Digital`) can be stored within the files. Both types can be selected, used as base for visualization, or their time stamp information edited from within this window. Use the `Start`-pop-up menu (figure 5←a) to select the packet-label which marks the beginning of a stimulus. If the same analog or digital data packet is used to mark start and end of a stimulation, adjust the `increment` between successive stimulations via the

[12]http://cybernetics.com/NEVspc20.pdf

text entry field in the same line. If a different data packet is used to mark the end of the stimulation, choose the corresponding data packet-label from the `End`-pop-pup.

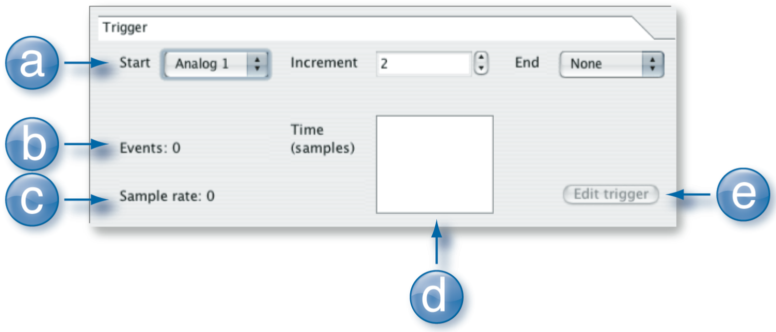


Figure 5: Options-Tab: Trigger

Information of the number of the selected trigger events is provided in figure 5←b. The `sample rate` at which these information was acquired during the experiment in figure 5←c. Figure 5←d displays a list of the timestamps of the selected trigger packet/s (timestamps are displayed with their sample number saved during the acquisition of the actual opened experiment). Up to 32756 different of these timestamps can be directly edited from within this tab by clicking the the `Edit Trigger`-button (figure 5←e).

Binning, ISI, Frequency `Binning`, inter spike interval (`ISI`), and `frequency` related parameter can be accessed from this window.

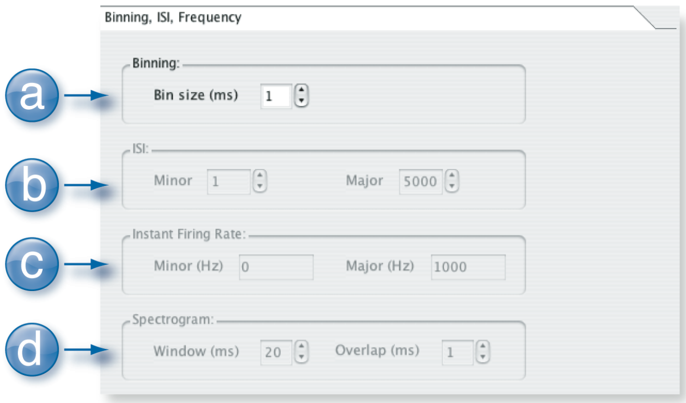
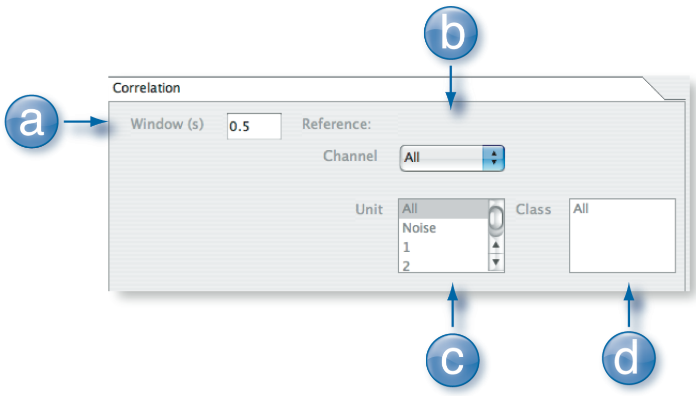


Figure 6: Options-Tab: Binning, ISI, Frequency

Use the widget figure 6←a to adjust the `bin size` to meet the requirements needed. Adjust the range within the inter spike intervals are extracted and displayed from the actual opened file using the `Minor`- and `Major`-widget in the window part marked as

←b. The frequency range for `instant firing rate` calculations can be adjusted with the widgets in figure 6←c. The `window length` and the `overlap` which is used, if a `spectrogram` calculation is carried out, can be adjusted via the widgets marked as figure 6←d.

Correlation This window allows the definition of all parameter for correlation-related calculations. Use the `Window`-text field marked in figure 7 with ←a to adjust the symmetrical time window in which the correlation function is determined. Set from the `Channel`-pop-up (7←b) the reference electrode and, if existing, the reference neuronal `Unit` (figure 7←c) or temporal response `Class` (figure 7←d) for the correlation calculation. Choosing `All` from the `Channel`-pop-up (7←b) means that the correlation of the selected electrodes is calculated against the population activity (cumulative activity of all recorded events exclusive the electrode/unit for which the correlation function is determined.)


Correlation

Window (s) 0.5

Reference:

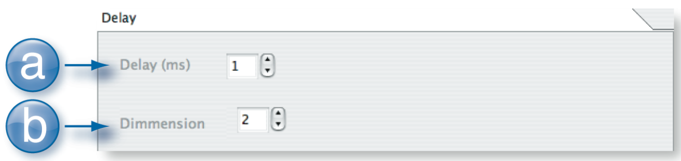
Channel All

Unit All, Noise, 1, 2

Class All

Figure 7: Options-Tab: Correlation

Delay Adjust the time lag (`Delay`, figure 8←a) and `Dimension` (figure 8←b) number which are used for delay embeddings. These parameter influence the phase space reconstruction of the opened or selected experimental data set. The method is taken


Delay

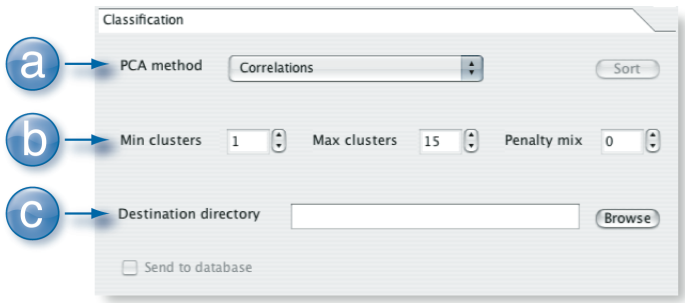
Delay (ms) 1

Dimmension 2

Figure 8: Options-Tab: Delay

from the TISEAN package and used in a few analysis functions of *neur*ALC.

classification *neur*ALC offers the option to classify within single or multiple experiments registered spikes according to their temporal response in the PSTH. The accessible parameter for this processing can be adjusted from this part of the `Options`-tab. Specify the eigenvector calculation using the pop-up-widget marked with ←a in figure 9.



PCA method: Correlations

Min clusters: 1 Max clusters: 15 Penalty mix: 0

Destination directory

Figure 9: Options-Tab: Classification

For the clustering process of this data some parameter can be adjusted via the widgets marked with ←b. The classification process can additionally applied to all experimental files in a directory defined in ←c in figure 9. In this case the actual defined parameter set in this and the other sub-windows of the `Options`-tab (e.g. the bin size, number of selected electrodes, etc.) is used for the classification of the directory data.

Selection A set of electrodes from the actual opened experiment can be selected. The data set represented by this selection is then used for all further analysis or visualization methods. The main part of the window is occupied by a 10x10 electrode matrix.

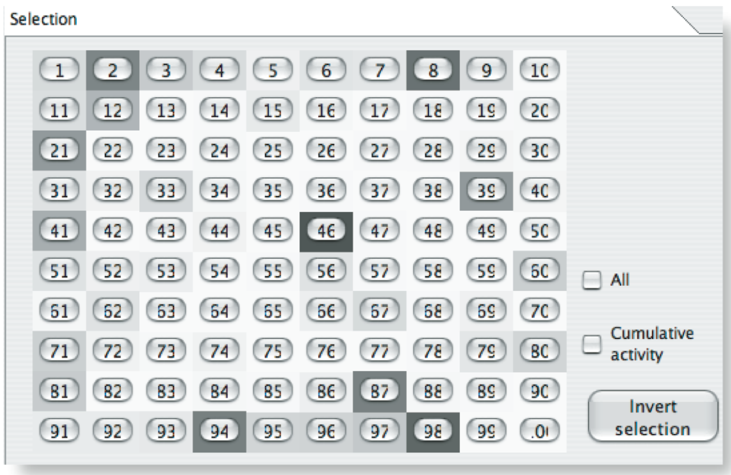


Figure 10: Options-Tab: Selection

When an experimental data file is opened the activity recorded on each electrode is color coded: darker button backgrounds mean high activity; lighter grey backgrounds lower or no activity. If the mouse cursor is shortly paused over one of the electrode buttons, a tool tip with the actual number of recorded events during this experiment on this electrode is shown. If the `All`-checkbox on the right becomes activated, all electrodes will be selected. The `Invert Selection`-button inverts the actual selection. Activating the `Cumulative activity`-checkbox will enable the calculation of the cumulative activity on all electrodes and, if appropriate, include it to data analysis and/or exported files.

“El Ultimo” This part of the `Options`-tab incorporates, as the name already indicates, access to function and/or parameter which do not really fit to any of the other option, or were implemented “last minute” to the program. This does not mean that this function were not comprehensively tested *well, as far as time permitted*- mainly the provided functionality is not fully implemented. For the first release of *neur*ALC the shuffling function (figure 11←a) offers just one way of doing it, and might be subject to further extension in the future. Likewise the function to extract the timestamps of a designated firing sequence of neuronal electrodes (figure 11←b), units or temporal response classes - the extraction works, but detection of existing fire pattern is not implemented so far. We have developed a small and simple external program to determine such fire patterns - if you interested in using this program and to collaborate, feel free to contact us.

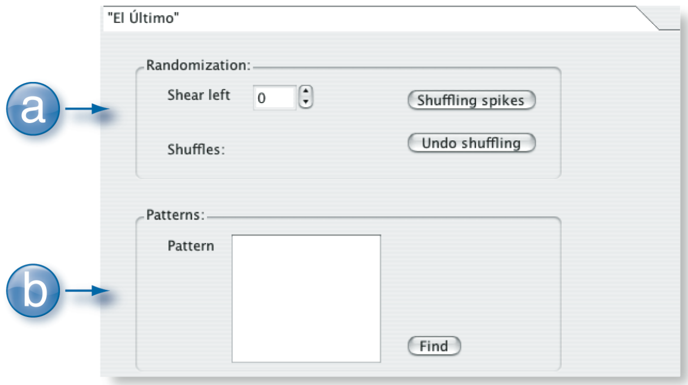


Figure 11: Options-Tab: “El Ultimo”

3.2 The *neur*ALC Menu Bar

The program menu bar (p.13, figure 12) provides access to all implemented analysis and visualization functions of *neur*ALC. If a command is chosen from any of the menus, all required parameters will be used as defined in this moment in the `Options`-tab (pp.8).

The menu bar is structured in 5 menus: the `neuralc`-, `File`-, `Analysis`-, `Contraptions`-, and `Help`-menu. Depending on the operating system *neur*ALC is running on, the menu, as shown in figure12, and its entries might look and behave slightly different. Many menu entries can be additionally navigated via keyboard short cuts - for a list of available keyboard commands for *neur*ALC see Appendix 7.1.

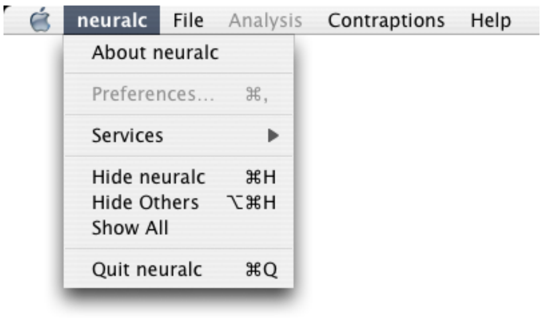


Figure 12: Menu bar: the neuralc-menu

3.2.1 The neuralc-menu

This menu (figure 12) provides access to the `About neuralc`-information and allows to quit the program.

3.2.2 The File-menu

From this menu experimental data files in NEV2-format can be opened and analyzed in *neur*ALC (`File`→`Open File`). Any opened files can be saved to a new file in NEV2-format (`File`→`Save to NEV2`).

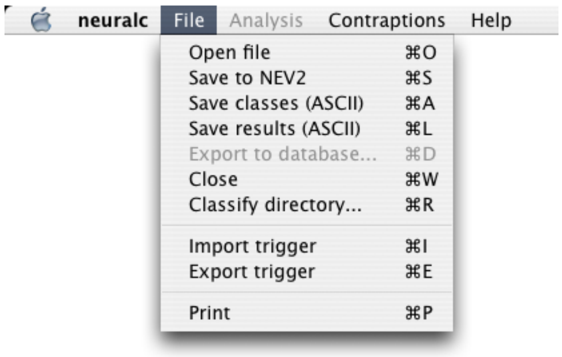


Figure 13: Menu bar: the File-menu

If the post-stimulus time data of the opened experiment were classified into TRCs,

the result can be save using the `File→Save classes (ASCII)`-entry from the menu. Calculation results can be saved to files in ASCII-format using the `File→Save Results (ASCII)` command. If MySQL was installed on the system in use, and successfully detected by *neur*ALC, the results of the actual displayed analysis can be transferred and stored in the database using the `File→Export to database...`-command. Choose the `File→Close`-command to close the actual opened and analyzed experimental recording. Additionally a directory which includes a set of experimental recordings can be specified and the recorded responses can be classified over all experiments according to their response characteristics in the post-stimulus time histogram (PSTH). The `Classify directory...`-command provides this functionality.
Via the `Import trigger`- and `Export trigger`-command it's possible to import an external file which includes "trigger" information about the applied stimulation, respectively to save this "trigger" information of the currently opened experiment to a file. *neur*ALC possesses a program-internal editor for the trigger information (p.8) - this editor is restricted in the size of the trigger information it can handle and does not offer highly sophisticated manipulation functions. Saving the trigger information allows to extract information from this data by using scripting or programming languages, or external text editor. Finally the `File`-menu provides printing functionality - any graph calculated and displayed by *neur*ALC can be printed via `File→Print`-command.

3.2.3 The Analysis-menu

Selecting any command from this menu applies the chosen analysis to the data set visible in the selected tab: if the `Population`-tab is active, the analysis is carried out on population level; if the `Electrode`-tab is active, the calculation is carried out on and visualized for the selected electrode or neuronal unit.

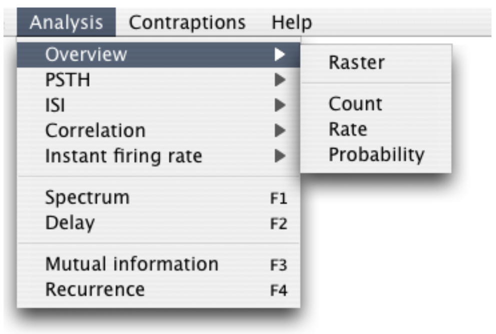


Figure 14: Menu bar: the Analysis-menu with the Overview-submenu selected

The Overview&PSTH-submenu (`Analysis→Overview/PSTH`) Choosing an entry from this menu (figure 14), applies the selected calculation on the selected data and visualizes the result over the whole experimental duration. Via this submenu it is possible to calculate and/or visualize the data as a raster plot (`Analysis`→`Overview`→`Raster`). The available additional options `Analysis`→`Overview`→`Count`, `Analysis`→`Overview`→`Rate` and `Overview`→`Probability` result in an overlay in which the number of spike per bin (`Count`), the firing rate per bin in Hz (`Rate`), or the probability per bin (`Probability`) is calculated and visualized. Identical options are available from the `Analysis`→`PSTH`-submenu resulting in similar forms of visualization for the post-stimulus time histogram data.

The ISI-submenu (`Analysis`→`ISI`) The commands from this submenu allow to calculate and visualize the inter-spike intervals (ISI) for the chosen data. Two additional options are available: the binned count (`Analysis`→`ISI`→`Binned count`), and the binned probability of the ISI (`Analysis`→`ISI`→`Binned probability`) can be calculated and displayed.

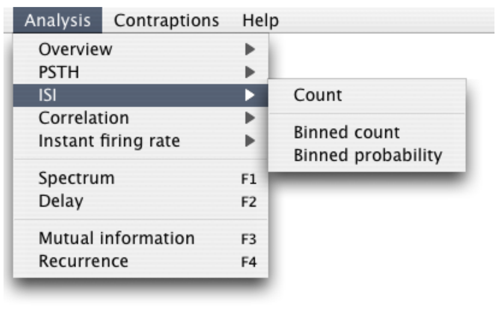


Figure 15: Analysis Menu: the ISI-submenu

The Correlation-submenu (`Analysis`→`Correlation`) This submenu allows to calculate the linear correlation or autocorrelation function of the actual selected data.

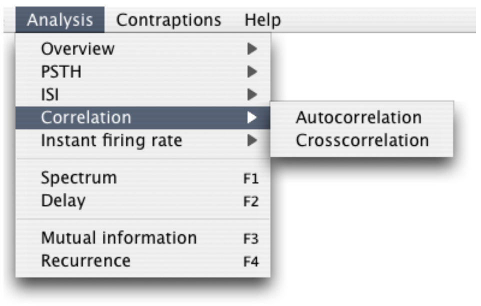


Figure 16: Analysis Menu: the Correlation-submenu

The Instant firing rate-submenu (`Analysis`→`Instant firing rate`) This submenu provides the functionality to calculate the instant firing rate on population or electrode level.

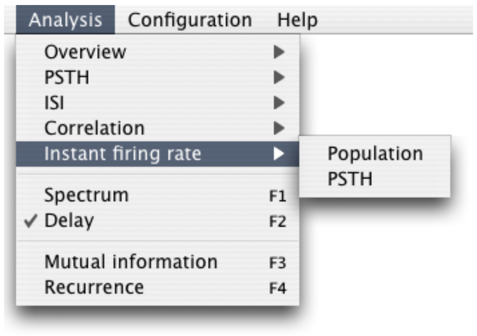


Figure 17: Analysis Menu: the Instant firing-menu

The Spectrum-submenu (`Analysis`→`Spectrum`) Linear time series method which computes a power spectrum by binning adjacent frequencies for the binned selected data. Implementation uses the FFTW3 library.

The Delay-submenu (`Analysis`→`Delay`) This function can be employed to unfold the multidimensional structure of the selected data. Delay calculations are computed according to the options given in the `Options`-tab. Implementation taken from the TISEAN package.

The Mutual information-submenu (`Analysis`→`Mutual information`) Estimates the time delayed mutual information of the data according to the options given in the `Options`-tab using a fixed mesh of boxes. Implementation taken from the TISEAN package.

The Recurrence-submenu (`Analysis`→`Recurrence information`) This function produces a recurrence plot of the, possibly multivariate, data set. That means, for each point in the data set it looks for all points, such that the distance between these two points is smaller than a given size in a given embedding space. Implementation taken from the TISEAN package.

3.2.4 The Contraptions-menu

This menu (figure 18) provides access to a variety of functions with different purpose: use the `Analysis`→`Contraptions`→`Save configuration`-command to save the actual parameter defined in the `Options`-tab to the configuration file `neuralc.conf`. This file is located inside the folder in which the *neur*ALC was installed and read at program startup. The `Contraptions→Select 3D color map`-command allows to load and use a user-defined color map for the visualizations in the `3D`-tab3.1.3.

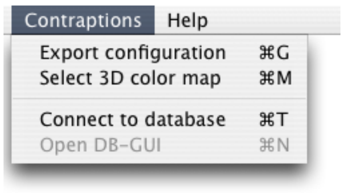
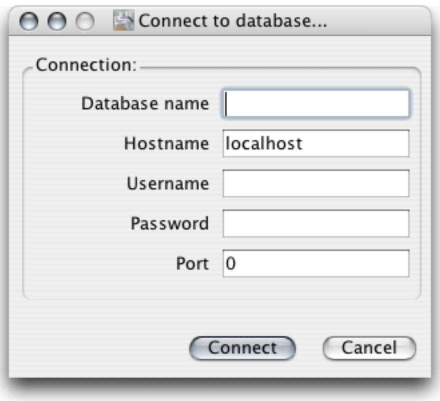


Figure 18: Menu bar: the Contraptions-menu

The `Contraptions→Connect to database`-command allows to connect to a MySQL-server to transfer and store experimental data files and results. Selecting the command opens a window in which the required information for the connection can be determined.


Connect to database...

Connection:

Database name	
Hostname	localhost
Username	
Password	
Port	0

Figure 19: Contraptions menu: the Connect to database-dialogue

Use the the text entry fields shown in figure 19 to define the name of the existing database, the host name on which the MySQL-server is running, the user name and password, and the port on which the database server allows connections. If *neur*ALC successfully connects to a MySQL database, the `Contraptions→Open DB-GUI`-command can be used to open the provided graphical database user interface (p.26).

3.2.5 The Help-menu

The help menu in the menu bar can be used to open *neur*ALC's online help.

4 Analysis & Visualization Background

4.1 *neur*ALC's Analysis Workflow

This section gives a short overview on the background of the different analysis methods used in *neur*ALC. The program incorporates methods taken from linear linear time series analysis (spectrum, auto- and cross-correlation, histograms, etc.) and low-dimensional chaos concepts (recurrence, delay, etc.), which has proven to be fruitful in the understanding of many complex phenomena (despite the fact that very few natural systems have actually been found to be low dimensional deterministic in the sense of the theory). It is of course up to the scientist who does the analysis to put these results into their proper context and to infer what information she or he may find useful and plausible. Figure 20 illustrates *neur*ALC's underlying workflow for analysis and visualization.

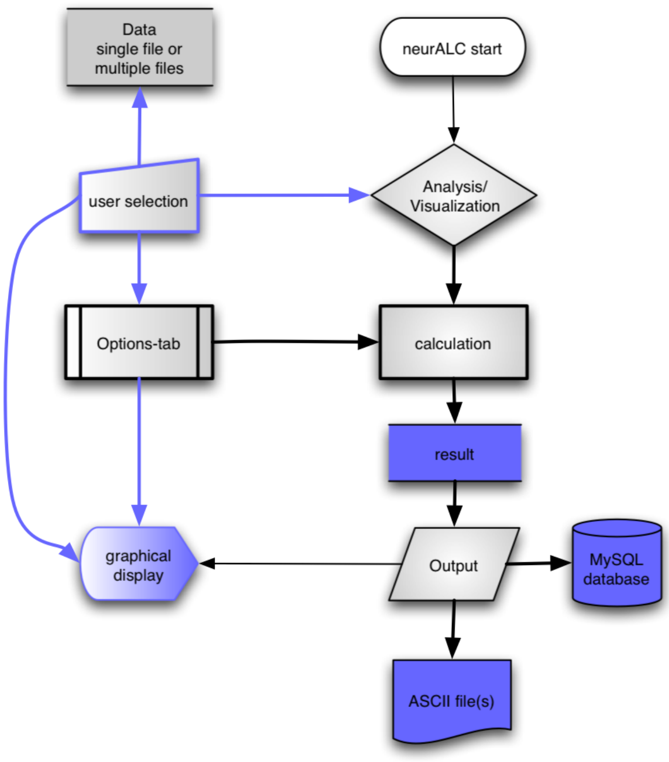


Figure 20: *neur*ALC's simplified analysis workflow

The user selects a single or multiple data files, predetermines all required parameter via *neur*ALC's `Options`-tab (p.8), and selects the desired analysis and visualization. The specified calculation is carried out and the result can be displayed, stored in an ASCII-file, or transferred and stored in a MySQL database. The different colored arrows represents the two, basic workflows in *neur*ALC: an "interactive" path based on the GUI, which allows exploration of a loaded data file, connected by the blue arrows. It is mainly intended to adjust and find the analysis parameter as needed, but allows as well to save individual visualizations, or the results for a selected ensemble of electrode-, neuronal unit-, or temporal response class-data.

If not explicitly saved using the `Contraptions`→`Export configuration`-command (p.16), changes to any parameter in the `Options`-tab (pp.8) are only valid and hence used during the actual *neur*ALC session.

Connected by the black arrows the “semi-automatic” path is shown , which uses the previously, interactively determined parameter and runs with minimal user interactions, executing the selected calculation for a set of data files, electrodes- or neuronal units-selection, and saves the computed results to external file(s) for further analysis. The specified calculations or visualization carried out in this “path” will use the required parameter as defined within the `Options`-tab (p.8) in the moment of analysis - this important conceptual aspect is elucidated by the thick, black stroke around the Options- and Analysis-box in figure 20.

All available calculations and visualizations can be carried out as well on population-, as on electrode-, neuronal unit-, or temporal response class level. Selected calculations or visualizations are therefore GUI context-sensitive: the visualization depends on the actual selected tab in the *neur*ALC GUI. .

4.2 Spike Train Representation

*neur*ALC offers a few ways of representing the recorded data. Basically data can be displayed on population- (p.5) or individual electrode-, neuronal unit-, or temporal response class-level (p.6). In this context spike data set can be displayed over the full time course of the opened experiment (`Analysis`→`Overview`, p.14), or displayed as post-stimulus time histogram (`Analysis`←`PSTH`, p.15).

Rasterplots&Activity Estimation

Overview As default experimental data is displayed in forms of a raster plot (figure 21). Each recorded spike event is displayed using a single black dot. Along the abscissa the experimental time course is represented, along the ordinate the active electrodes (labeled as `Channel`), which logged the neuronal activity, are shown. Using the left mouse button it is possible in any 2D-display to zoom inside the data plot - figure 21, for example, shows the recorded activity on 100 active electrodes for the time interval between the 15th till 30th second of the experiment using this functionality. For this example a periodic stimulation with a period length of approx. 2s was used, resulting in the correlated neuronal firing pattern shown in the figure.

Selecting the `File`→`Print`-command allows to print the active data display inside *neur*ALC. If a particular region of the data representation was magnified (as shown in figure 21) using the mouse-based Zoom-function, only this area of the plot will be printed (or, if this functionality is supported by the OS, saved to a file.) .

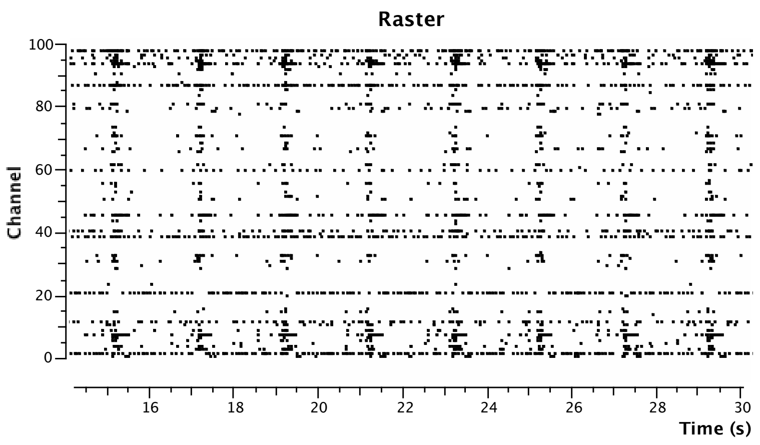


Figure 21: population raster plot

Post-Stimulus Time Histogram (PSTH) Another way of characterizing the recorded neuronal response pattern is provided in forms of the post-stimulus time histogram. Using the time t of the stimulus, the spikes are displayed in a window $\Delta\tau$ aligned at t.

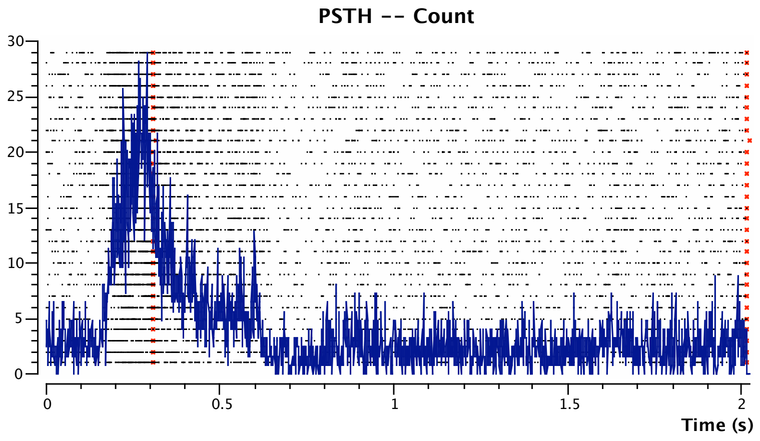


Figure 22: population PSTH plot

The whole experimental recording shown partly in figure 21 is plotted in this way in figure 22. The black dots represent each recorded event, along the abscissa the time interval $\Delta\tau$ aligned to the occurrence of the stimulus at t, marked by the red small crosses, is shown. Along the ordinate the each presentation of the stimulation is represented. Depending on the information provided within the experimental data file, $\Delta\tau$ can easily adjusted using the functionality provided in the `Trigger`-options (p.8).

Activity Estimation Overlays *neur*ALC provides 3 methods to estimate the population- or electrode/neuronal unit-activity from the data shown in the raster plots. By applying a binning procedure it is possible to calculate the simple count (spikes/bin), rate (in Hz), or probability/bin of the recorded neuronal events (see. pp.14). The result of the selected calculation is then displayed as an overlay in the actual rasterplot, exemplary shown in figure 22 for the spike count (superimposed to the raster with a blue line; bin size = 1ms).

ISI This function (p.15) allows to calculated and display the interspike interval (ISI) distribution for the population or the PSTH data of the selected electrode-, or neuronal unit-data. Only

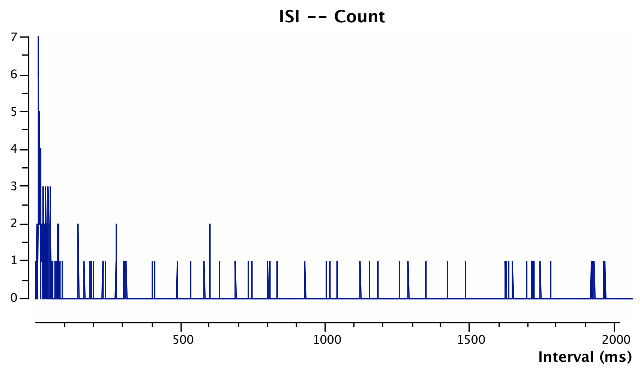


Figure 23: example of a PSTH-based ISI distribution of a single electrode

interspike intervals of the actual data set which match the limits defined in the related `Options`-tab (p.9) are extracted and used to calculate the interval distribution.

Instant Firing Rate Calculates the instant spike firing rate $r(t)$ for the population-, or selected electrode- or neuronal unit-data (p.16). The original spike time data is divided into bins,

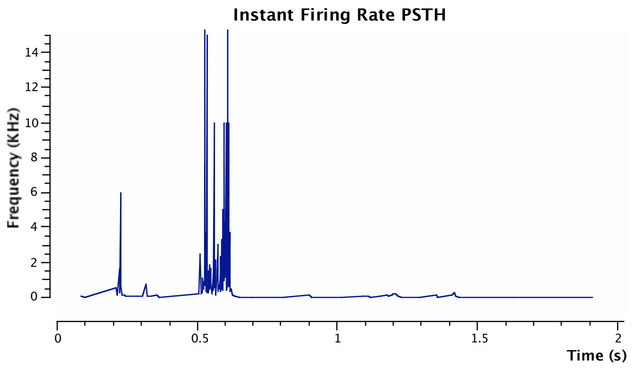


Figure 24: instant firing rate function for the PSTH data of a single electrode

each of width $\Delta\tau$. Thus if one looks on a bin centered on time t, a function $n(t)$ can be defined with $N = 1$ if there is a spike in the bin and $n = 0$ if there is not. Whether a particular spike t_i

is in a bin with size Δt at time t can to such an extent evaluate. To find any spike in the defined bins all possible spikes have to be summed, so that the function $n(t)$ that counts the spikes can be written as $n(t) = \sum_i f[\frac{t-t_i}{\Delta\tau}]$. From the sum in each bin then the firing rate in Hz in each is calculated and plotted.

Spectrum Computes a power spectrum of the binned data by binning adjacent frequencies. This routine uses the FFTW3 library and by default zero padding, that is, no periodic continuation is assumed.

4.3 Multivariate Time Series Analysis Methods

4.3.1 Correlation

Computes the auto/cross correlation function of a binned spike time series (p.15).

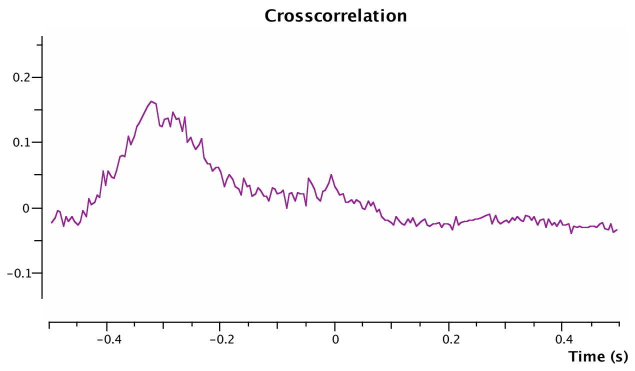


Figure 25: example of a cross correlation function of a single electrode

This function computes the cross correlations between two binned time series. Auto or cross correlation functions are calculated symmetrically for the the window length defined in the `Options`-tab (p.15). The default value is 0.5s, so that the cross correlation function shown in figure 25 is calculated and visualized within the time window [-0.5:0.5]s.

4.3.2 Recurrence

Recurrence plots are a useful tool to identify structure in a data set in a time resolved way qualitatively. This can be intermittency (which is detectable also by direct inspection), the temporary vicinity of a chaotic trajectory to an unstable periodic orbit, or non-stationarity. They were introduced in [Eckmann 1987, Casdagli, 1997], where you find many hints on how to interpret the results. For each point in the data set it looks for all points, such that the distance between these two points is smaller than a given size in a given embedding space (p.10). The time series, e.g. the estimated population activity as binned count, is simply scanned and each pair of the time indices (i, j) whose corresponding pair of delay vectors has a distance $\leq\epsilon$ is marked with a dot. Thus in the (i, j)-plane, dots indicate closeness. In an ergodic situation, the dots should cover the plane uniformly on average, whereas non-stationarity expresses itself by overall tendency of the dots to be close to the diagonal. Of course, a return to a dynamical situation the system was in before becomes evident by a high dotted region far away from the diagonal. To achieve an agreeable display speed for the result of this analysis, the number of points calculated is decimated for the on-screen visualization. The implementation in neurALC is taken from TISEAN.

4.3.3 Delay

The time evolution of spiking neuronal system in some phase space $\Gamma \subset \mathbf{R}^d$ can be expressed in discrete time $t = n\Delta t$ by maps of the form $x_{n+1} = f(X_n)$. Neuronal recordings can then be thought of as a sequence of observations $\{s_n = s(X_n)\}$ performed with some measurement function $s(\cdot)$. Since this scalar sequence $\{s_n\}$ in itself does not properly represent the multidimensional phase space of the dynamical system, this function can be employed to unfold the multidimensional structure of the available data. The method of delays represents one of the most important technique for phase space reconstruction. Vectors in the embedding space, are formed from the time delay values of the scalar measurements $s_n = (s_{n-(m-1)\tau}, s_{n-(m-2)\tau}, ..., s_n)$. The number m of elements is called the *embedding dimension*, the time τ is generally referred to as the *delay* or *lag*. The implementation in neurALC is taken directly from TISEAN.

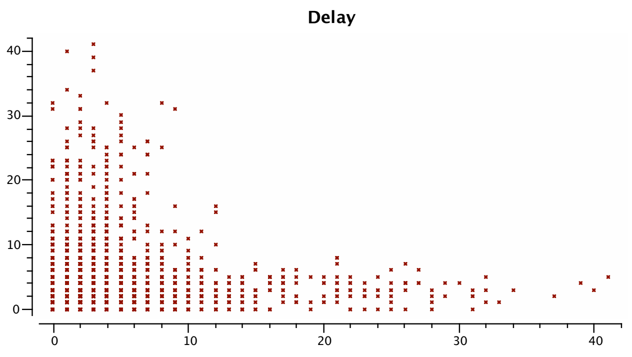


Figure 26: delay representation of a population recording (delay=25ms, dimensions=2)

4.3.4 Mutual Information

This function estimates the time delayed mutual information [Fraser and Swinney, 1986] of the binned selected data using a fixed mesh of boxes. Unlike the autocorrelation function, the mutual information takes in account also nonlinear correlations, and is intended to determine a reasonable *delay* (p.10). Computing $S = \sum_{ij} p_{ij}(\tau) \ln \frac{p_{ij}(\tau)}{p_i p_j}$ where for some partition on the real numbers p_i is the probability to find a time series value in the i-th interval, and $p_{ij}(\tau)$ is the joint probability that an observation falls into the i-th interval and the observation time τ later falls into the j-th. For embedding dimension ≤ 2 exist good arguments that if the time delayed mutual information exhibits a marked minimum at a certain value of τ, then this a good candidate for a reasonable time delay. The implementation in neurALC is taken from TISEAN.

4.4 3-D Visualization

*neur*ALC offers some visualization functions in three-dimensional space. Select the 3D-tab (p.7) and choose `PSTH` from the `Option`-tab to visualize the cumulative binned PSTH of the electrode set defined in the global `Options`-tab (p.11) in 3D (figure 27). As in the 2D-display (p.15) the PSTH interval is represented along the abscissa; the ordinate shows the number of stimulus representations chronologically ordered from top to bottom. The binned spike number is displayed using a user-definable color map (p.7). Using the mouse the data view can be rotated, zoomed, or panned (see full list of keyboard short cuts & mouse navigation options pp.35).

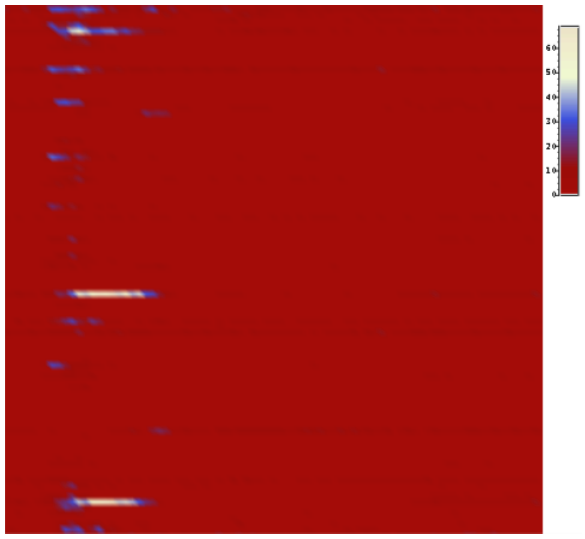


Figure 27: default view of a binned cumulative PSTH visualized in 3D

Additionally two forms of a spectrogram can be calculated and plotted: for the full experimental course, or for the PSTH data of the actual selection (use the `Option`-pop-up menu in the `3D`-tab (p.7) to select the desired visualization). To calculated the spectrogram, the binned

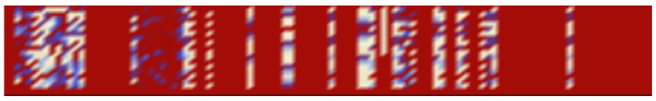


Figure 28: default view of a spectrogram calculated for the PSTH of a selected electrode

neuronal response is divided in windows with a preset length and overlap (p.9). Next a fourier transform (FFT) is applied to the data in each window, deriving the frequencies and amplitudes of its component simple waves. Depending on the size of the fourier analysis window used for the FFT analysis, different levels of resolution can be achieved. A long window resolves frequencies at the expense of time - the result is a narrow band spectrogram, which reveals individual component frequencies. If a small analysis window is used, adjacent frequency components are smeared together, but the result displays a better time resolution. Spectrograms as shown in figure 28, represent the the time along the abscissa. The “time” unit used is the number of the window which is used to derive the frequency components, which are then represented along the ordinate axis. The amplitudes of the frequencies are displayed color coded in z. For the moment this visualization technique is intended only as a qualitatively measure.

4.5 Classification of electrode&“unit” responses

4.5.1 Concept of Temporal Response Type Classes (TRC)

*neur*ALC offers a method to classify the recorded neuronal response pattern according to their time course in the PSTH. The resulting classes are denominated as *temporal response class* (TRC). Such a class describes the the binned time course of the response as calculated in the post-stimulus time histogram. Similar binned responses are assigned to the same TRC. To manage large data sets of neuronal recordings and to represent this data, the program uses the following sequential approach to classify the responses into temporal response classes:

1. Reducing the dimensionality of the binned post-stimulus time histogram data using Principal Component Analysis (*PCA*).
2. Cluster analysis (based on KlustaKwik) which reduces the number of the objects resulting from the first step by placing them into newly assigned *TRCs*.

If you’re searching for a program which does “classical” “unit”-sorting based on the waveform of the recorded spike signals, you might take a look on another free program, NEV2lkit, which was developed in our lab. NEV2lkit is a spike preprocessor, which allows the extraction of spike signals from continuous recorded data and/or neuronal spike classification in accordance with their signal shape (wave form). NEV2lkit is additional recommended, if you want to convert ASCII-based, or e.g. NEV1.x files to NEV2.0 for further usage with *neur*ALC.

Principal component Analysis (PCA) The estimated activity signal in the PSTH represent a large number of variables which makes it very likely that subsets of these variable are highly correlated to another. The accuracy and reliability of a classification of such data will suffer if highly correlated or variables which are unrelated to the outcome are included to the analysis.

Reducing the dimensionality of the data in the PSTH without sacrificing accuracy is therefore a key step in this classification. PCA transforms a number of (possibly) correlated variables into a smaller number of uncorrelated variables - the principal components. As a result the dimensionality of the dataset is reduced but most of the original variability retained. The first principal component accounts for as much of the variability in the data as possible, and each of the succeeding components accounts for as much of the remaining variability as possible.

*neur*ALC performs Principal Component Analysis on the PSTH data constructed for each selected file and its contained electrode or neuronal “unit” data. It calculates the first 3 principal components which are then used in subsequent cluster analysis based on KlustaKwik. Using the related pop-up menu from the `Classification`-window in the global `Options`-tab in the GUI (p.11), it is possibly to specify whether the eigenvector calculation is based on the *correlations*, the *variance/covariances* or the *sum of squares of the cross-products* of the binned PSTH signal matrix.

Clustering Last step of the unit sorting is the clustering of the data. *neur*ALC uses KlustaKwik, a program for unsupervised classification of multidimensional continuous data. KlustaKwik delivers among others: fit a mixture of Gaussians with unconstrained covariance matrices, chooses automatically the number of mixture components, and runs fast on large data sets. KlustaKwik is based on the classification expectation maximization algorithm (CEM) [Celeux and Govaert, 1992]. Definable parameters for the clustering process in the *neur*ALC GUI are:

MINCLUSTERS the random initial assignment will have no less than n clusters. The final number may be different, since clusters can be split or deleted during the course of the algorithm. The default value is 1.

MAXCLUSTERS defines the maximum of possible clusters n. Cluster splitting can produce no more than n clusters. The default value is 5.

PENALTYMIX it is possible to specify Bayesian information content (BIC) or Akaike information content (AIC) as the penalty for a larger number of clusters or a mixture of these two. This widget allows to define the amount of BIC to use a penalty for more clusters. Default of 0 sets to use all AIC. Use 10 to use all BIC (this generally produces fewer clusters).

All other parameters which are known in the standalone version of KlustaKwik program can be edited in the source code of *neur*ALC.

5 MySQL Database Integration

*neur*ALC provides some database functionality using MySQL. MySQL is an open source SQL database which, quoting the MySQL website, is “designed for speed, power and precision in mission critical, heavy load use.” Within the MySQL database original experiment files, analysis results and the information of the appendant parameter used, modified data, comments, or even digital photographs of the neuronal specimen, which was used in the experiments, can be stored and administered. *neur*ALC integrates seamlessly with the MySQL client - results of calculations, comments, original experimental data files and any modification, etc. are tracked and automatically transferred to the database. A second, supplementary program - *db*ALC - is provided. It offers a simple graphical user interface to access, administer and retrieve data stored in the MySQL database.

5.1 Technical Prerequisites

To use *neur*ALC’s database functionality, an installation of MySQL version 3 or higher is needed. Please visit `http://www.mysql.com` and download the package of your choice. *neur*ALC 1.0.0 was successfully tested with MySQL 3, 4 and 5.
Additionally the Qt3 libraries on your system must offer MySQL-driver support (the supplied binary installers of *neur*ALC for Apple’s MacOSX and Microsoft Windows include

the Qt3 libs compiled with the required support. Under Linux an update via the package manager of your distribution, or even a recompilation of the Qt3 C++ framework might be needed).

Install MySQL on your computer system - from our experience, depending on the operating system and the package of MySQL you choose, this might be the most problematic part. Before you try to use the database functionality provided by *neur*ALC or *db*ALC, make sure that you check the correct function of the MySQL installation on your system. Refer for the MySQL information which comes with the package you downloaded, or the online documentation available at `http://www.mysql.com/documentation`.

If MySQL is successfully installed and running, add a user "neuralc" with the password "neuralc" to the database (refer to the MySQL documentation for information about adding a user and password). This is the database account which is used as default from within *neur*ALC. In the next step a database with the required design and table type "InnoDB" must be created. The easiest option is to copy the files shown in figure 29 (provided with the *neur*ALC distribution inside the folder `neuralcDB`) into MySQL's default folder for databases. The next time you startup MySQL these files are found

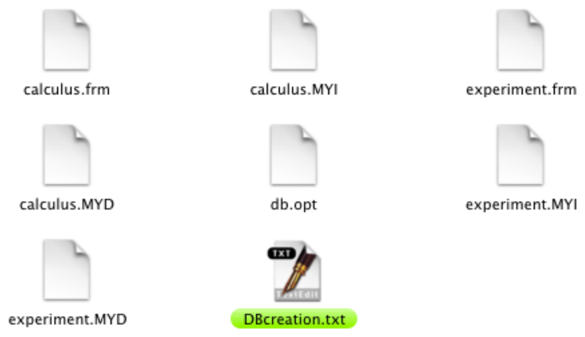


Figure 29: Content of the folder `neuralcDB`

and can be accessed. Alternatively you can open the file `DBcreation.txt` (figure 29, labeled in green) in the editor or text processor of your choice, and use the documented command sequence from within MySQL to create the correct tables and keys (the required command log is as well included in section 5.4, p.30).

5.2 Database Design

The database design which is used by *neur*ALC is sketched in figure 30. While it is surely not the most sophisticated database design, it'll *hopefully* suits the "minimal" requirements.

Two main tables, `experiment` and `calculus` are employed. `experiment` is related with the fields in which the original experimental data file in NEV2-format (`nev_file`), a comment (`comment`), the creation data (`creation_date`), a digital photograph (`photo`), the experiment related stimulus program e.g. a Python script (`stimulus_program`), or the file name (`file_name`) is stored. In short these tables hold all experiment related information apart from the information which is related to any applied analysis.

Analysis results become represented in the `calculus` entitity and related tables (marked in the figure by a dark grey background), which hold this analysis related information.

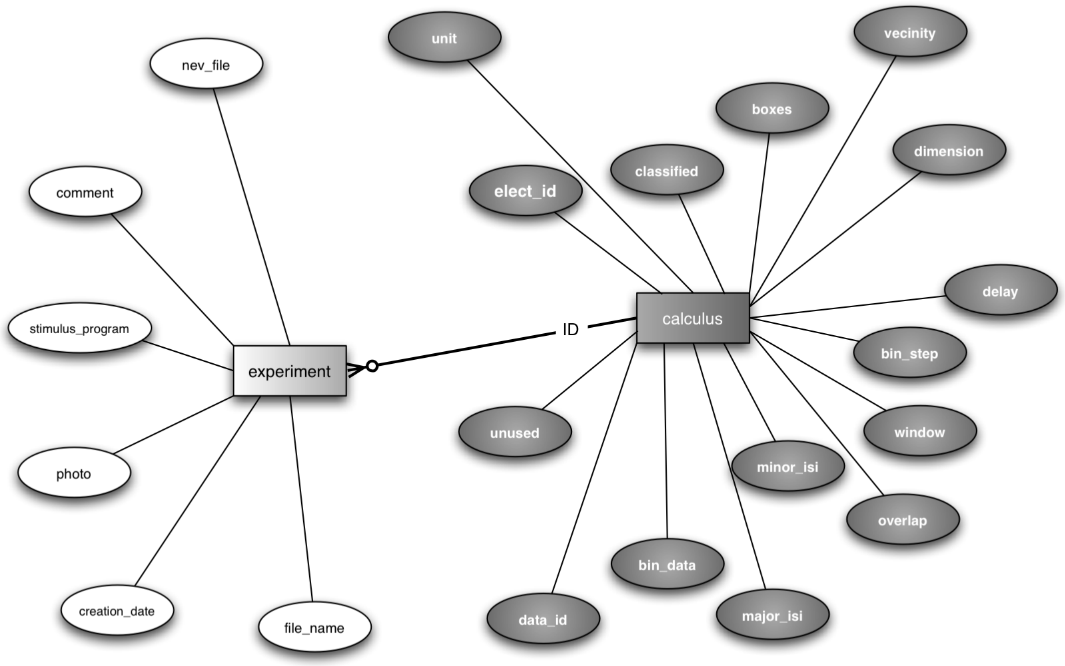


Figure 30: Extended Entity Relation Design of the database used by *neur*ALC

Different tables for storing the results and related analysis parameter are created: the electrode number (`elec_id`), if available the unit information (`unit`), the status of the TRC classification (`classified`), recurrence related information (`boxes` and `vecinity`), the result of the `delay` calculation, the embedding `dimension` used, the applied bin step width (`bin_step`), the frequency analysis related information about the used `window` size and possible window `overlap` in this calculation, the ISI-related information (`minor_ISI`, `major_ISI`), the result of an applied binning (`bin_data`), and the general data information (`data_id`). Additionally one, so far unused key (`unused`) is created.

5.3 *db*ALC - a database GUI

While experimental data files and related results can be transferred to the MySQL database, an additional program *db*ALC is intended for more convenient data retrieval. It provides a simplified graphical user interface (figure 31, p.29) to browse the data, and retrieve the stored data.

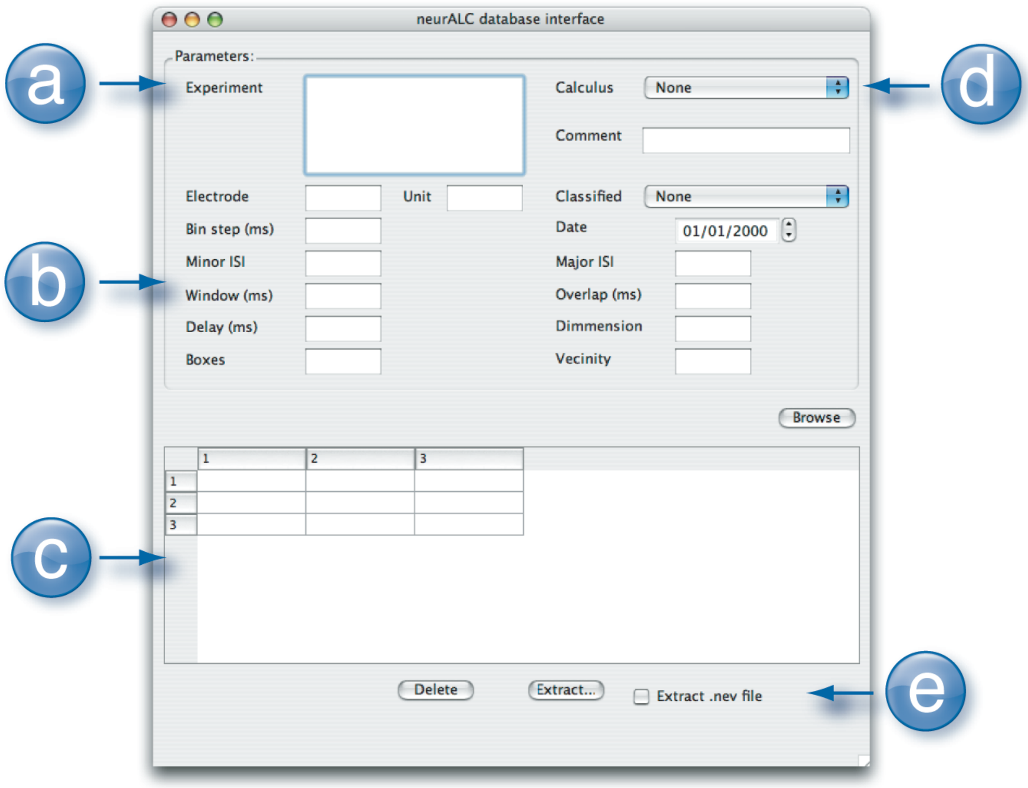


Figure 31: The *db*ALC GUI

The *db*ALC can be divided in five functional areas, labeled `a`-`e` in figure 31. A listview displays all actually saved experiments within the *neur*ALC database. Any experiments can be selected from this window. The pop-up menu labeled with `Calculus` (`d`) allows to select the specific type of analysis result which is available from this experiment. Furthermore any selection from this pop-up will dynamically change the list of experiments displayed in `a`. This means that, if for example PSTH is selected from the pop-up, the list of experiments which can be selected, becomes restricted to the ones which include this type of analysis result.
The text entry fields in `b` display the related parameter information, or allow to enter directly a specific parameter, for which then the experiments, which contain this information, will be filtered from the database and displayed in the listview (`a`).
Analysis results will be displayed in the table view labeled with `c`. In the column-view the stored results for each, for the defined analysis chosen electrode or neuronal unit (see p.11) will be displayed. The selected analysis results can be extracted (or deleted) using the `Extract...`-button in `e`. Activating the `Extract .nev file`-checkbox will additionally extract the corresponding experimental data file in NEV-format from the database.

5.4 SQL command log for creating the *neur*ALC database

```
CREATE DATABASE neuralcDB;

GRANT SELECT,INSERT,UPDATE,DELETE
ON neuralcDB.*
TO 'neuralc'@'localhost'
IDENTIFIED BY 'neuralc';

FLUSH PRIVILEGES;

CREATE TABLE experiment (file_name VARCHAR(256),
nev_file LONGBLOB,
comment VARCHAR(256),
creation_date DATE,
photo LONGBLOB,
stimulus_program LONGBLOB,
PRIMARY KEY(file_name(256)));


CREATE TABLE calculus (data_id ENUM(
'None',
'PopulRaster',
'PopulCount',
'PopulRate',
'PopulProb',
'PSTHRaster',
'PSTHCount',
'PSTHRate',
'PSTHProb',
'ISICount',
'ISIBinCount',
'ISIBinProb',
'IFRPopulation',
'IFRPsth',
'Autocorr',
'Crosscorr',
'Spectrum',
'Delay',
'Mutual',
'Recur',
'PSTH3D',
'SpecPopul3D',
'SpecPSTH3D'),
elect_id SMALLINT(3),
unit SMALLINT(3),
classified TINYINT(1) UNSIGNED,
boxes SMALLINT(4) UNSIGNED,
vecinity SMALLINT(4) UNSIGNED,
bin_step SMALLINT(4) UNSIGNED,
delay SMALLINT(4) UNSIGNED,
window FLOAT UNSIGNED,
overlap SMALLINT(4) UNSIGNED,
dimmension SMALLINT(4) UNSIGNED,
minor_isi SMALLINT(5) UNSIGNED,
major_isi SMALLINT(5) UNSIGNED,
bin_data LONGBLOB NOT NULL,
unused SMALLINT(5),
experiment VARCHAR(256),
FOREIGN KEY (experiment(256)) REFERENCES experiment ON DELETE CASCADE,
PRIMARY KEY (experiment(256),data_id, elect_id, unit,classified, boxes,vecinity, bin_step,delay,
 window,overlap, dimmension,minor_isi,major_isi, unused));
```

6 A Real World Example

This section provides a short “real world” example. When opening an multi-electrode recording in NEV2-format, *neur*ALC displays as default the opened data in the `Population`-tab, offering an overview of the experiment. Additionally the population activity is estimated (using the actual, in the `Option`-tab defined, bin size) and displayed as a blue lined `Count`-overlay, representing the number of spikes counted in each bin.

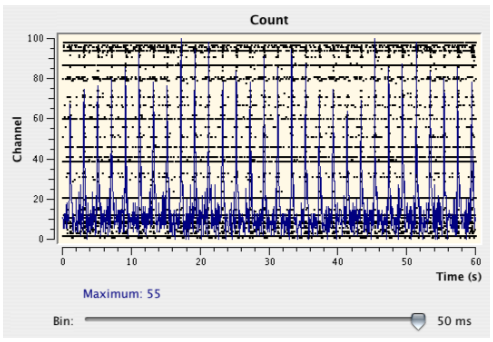


Figure 32: Rasterplot of full experimental time course and population activity overlay

In the lower part of the window, the maximum of the bin count observed in the opened experimental data file is displayed (`Maximum:55` in figure 32). The slider widget at the button can be used to interactively adjust the bin size - the population activity overlay will be immediately recalculated and the display updated. Using the menu-bar provided analysis function (p.14) the population activity can also be displayed in terms of its frequency or probability.

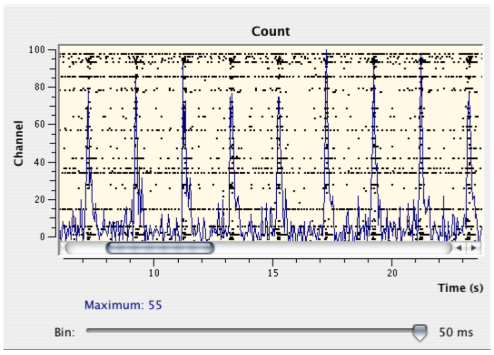


Figure 33: Zoomed population activity

Using the mouse it is possible to zoom the display (see section 7.2). For the example

shown here, the zoomed display allows easy visual inspection of the periodic modulation of the population activity. Clicking with the left mouse-button inside the display area sets a label with the time and channel coordinate.
Choosing the `recurrence`-function (p.22) a recurrence plot for the recorded neuronal activity in this experiment is displayed (figure 33). Lighter shaded regions in this plot indicate that these data are dynamically distinct from the rest (see p.22 for more information). In the lower part of the window the bin size used for the calculation is displayed (`bin size: 50ms`, taken as defined in the related window in the `Options`-tab). The slider widget at the bottom can be used to interactively vary the `vicinity`-parameter.

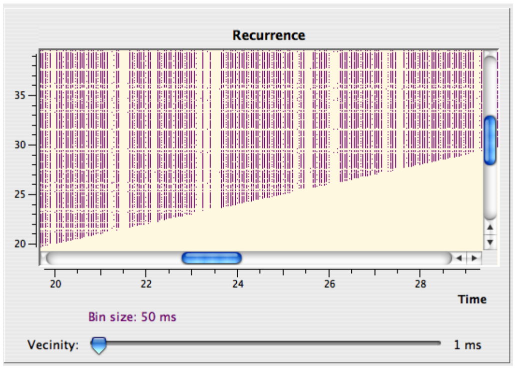


Figure 34: Zoomed recurrence plot for the binned population activity.

The underlying periodicity can be revealed by calculating the autocorrelation function (p.22) for the binned population activity. From the visual inspection in figure 32 and 33 the window length for the autocorrelation calculation was adjusted to 2.25s in the related window of the `Options`-tab (p.10).

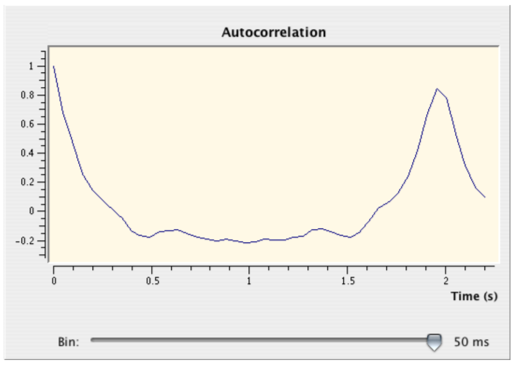


Figure 35: Autocorrelation function of the binned population activity.

The autocorrelation function shows a maximum at approx. 2s (peak amplitude is located at 1.998s, identified by simply clicking the tip of the peak at the right in figure 35). We now switch to the `Electrodes`-tab (p.6) and select the Rate-entry from the `PSTH`-command in the `Analysis`-menu (pp.14). Figure 36.

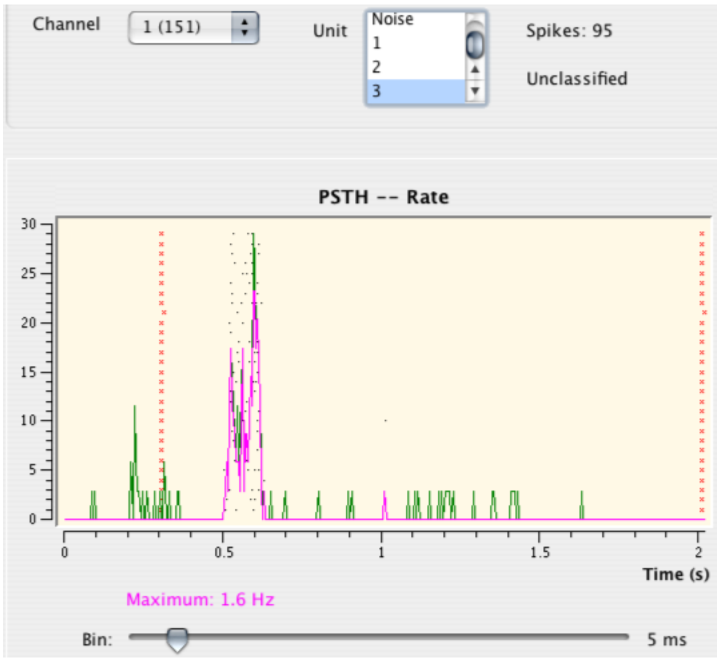


Figure 36: Post-stimulus time histogram for Electrode 1 with `Rate`-overlay

From the upper left `Channel`-pop-up the first electrode in this file was chosen. This pop-up displays the actual electrode selection, as well as the total number of spikes recorded during the actual experiment on this electrode - in this example: 151 spikes on electrode 1 - `1 (151)`.
The spikes of the experimental data shown in this example were previously sorted into "units" according to their signal form using NEV2lkit, a spike data preprocessor developed in our laboratory. If such "unit" sorting was applied, the `Unit`-listview in the upper right displays a list of the sorted "units" on this electrode. In the screen shot in figure 36 unit 3 was selected (marked by the light blue background in the list). Right to the listview the number of spikes, 95, associated with this unit is shown. Below this information a text field informs about the status of an additional classification of the temporal responses of these units (here "`Unclassified`"; see pp.25).
The 2D-plot area of the window shows the PSTH of the recorded spikes on this electrode as a rasterplot. The small red crosses in the display area mark the trigger information (see p.8) of the stimulation applied during this experiment. The green line overlay shows the estimated electrode activity for the PSTH data. The pink line overlay, which partly masks the green line, represents the activity of the selected "unit". The slider widget at the bottom can be used to change the bin size, and with it interactively

the estimated electrode, and selected “unit” activity. Using the `ISI`-command from the `Analysis`-menu (p.21) the inter-spike interval distribution for the selected can be visualized.

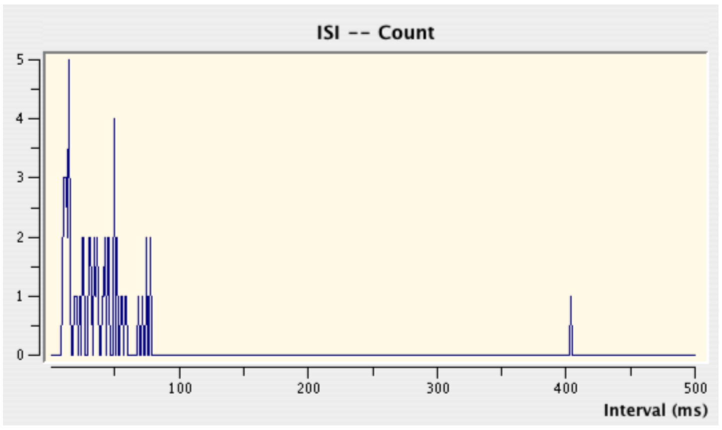


Figure 37: ISI distribution for the selected neuronal “unit”

In the next step the crosscorrelation of this neuronal “unit” with the binned population activity is visualized in the next figure (bin size = 1).

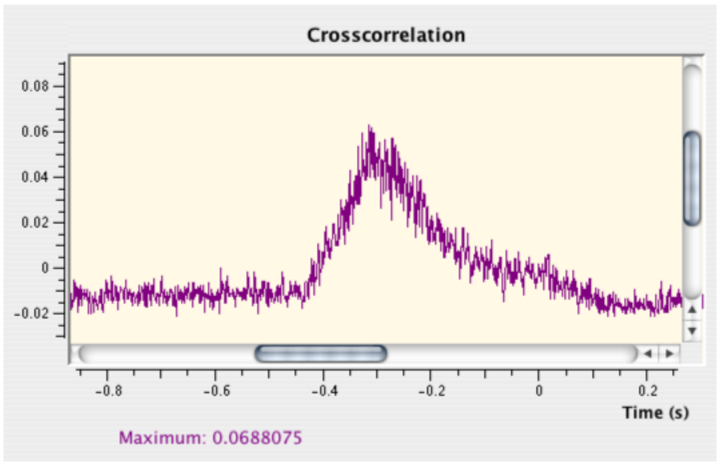
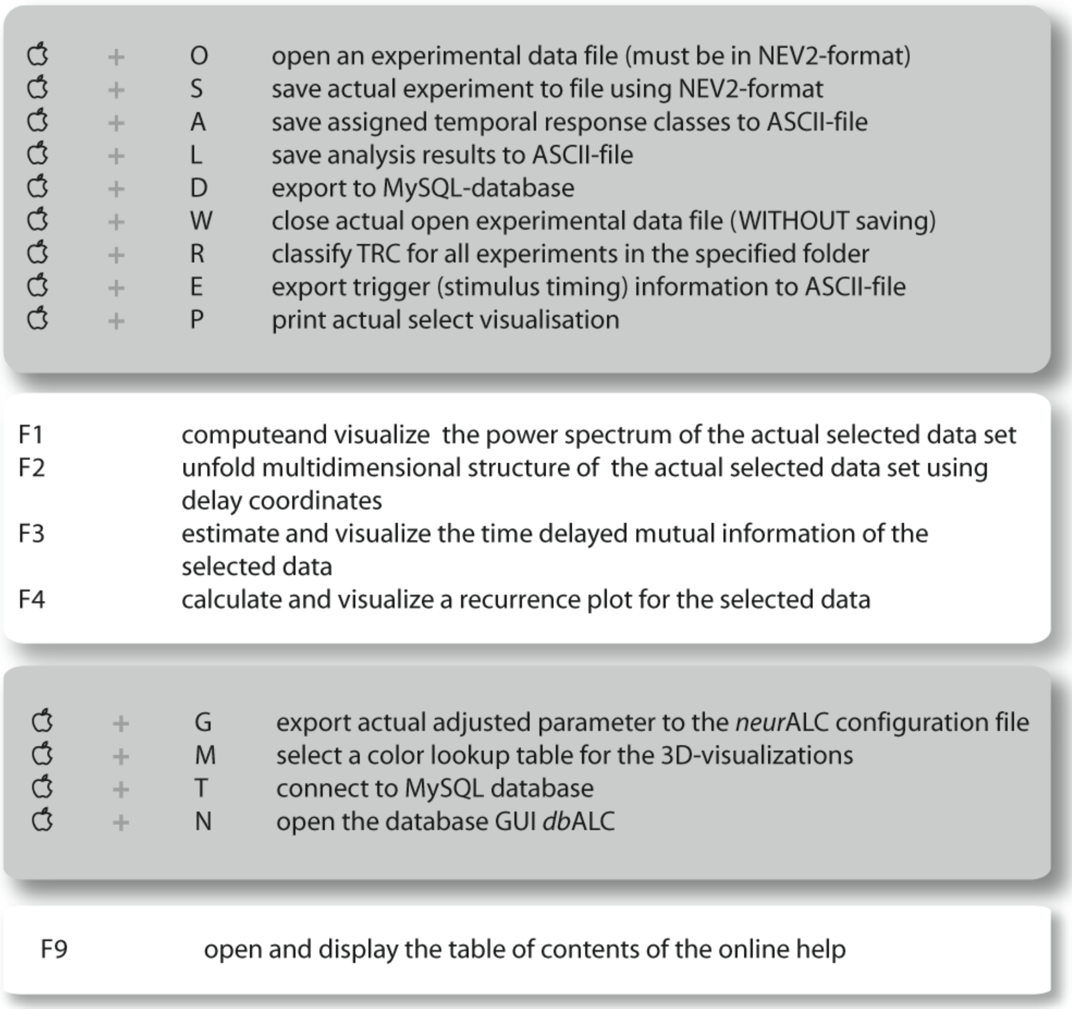


Figure 38: Crosscorrelation function for Electrode 1, Unit 3 with population activity

In the lower part of the window the maximum if the calculated correlation function is shown (this value is highly depending on the actual bin size used for the calculation!). To export the analysis result to an ASCII-file for further usage, use the `Save Results (ASCII)`-command from *neur*ALC’s `File`-menu (p. 13). If you selected previously a group of electrodes in the `Selection`-window of the `Options`-tab (p.11) the selected analysis was (or will be) carried out for the related data set. Accordingly the results of the analysis for all the selected electrodes and, if existing, neuronal “units” or temporal response classes will be exported and saved in the ASCII-file.

7 Appendix

7.1 Keyboard Short Cuts



⌘	+	O	open an experimental data file (must be in NEV2-format)
⌘	+	S	save actual experiment to file using NEV2-format
⌘	+	A	save assigned temporal response classes to ASCII-file
⌘	+	L	save analysis results to ASCII-file
⌘	+	D	export to MySQL-database
⌘	+	W	close actual open experimental data file (WITHOUT saving)
⌘	+	R	classify TRC for all experiments in the specified folder
⌘	+	E	export trigger (stimulus timing) information to ASCII-file
⌘	+	P	print actual select visualisation

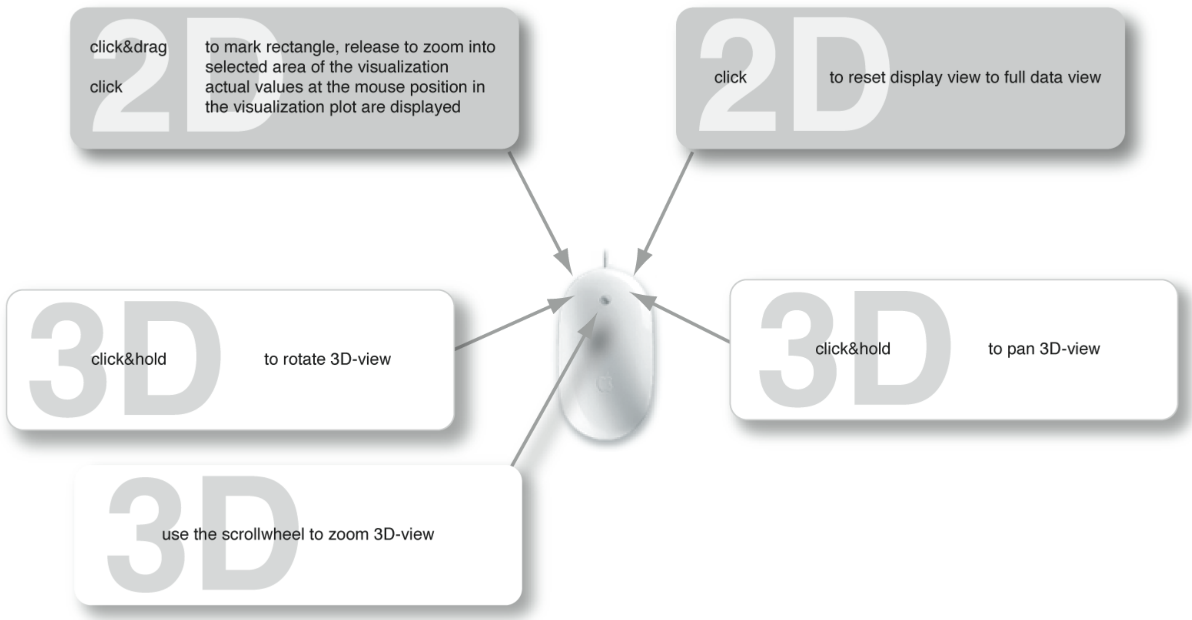
F1	computeand visualize the power spectrum of the actual selected data set
F2	unfold multidimensional structure of the actual selected data set using delay coordinates
F3	estimate and visualize the time delayed mutual information of the selected data
F4	calculate and visualize a recurrence plot for the selected data

⌘	+	G	export actual adjusted parameter to the *neur*ALC configuration file
⌘	+	M	select a color lookup table for the 3D-visualizations
⌘	+	T	connect to MySQL database
⌘	+	N	open the database GUI *db*ALC

F9	open and display the table of contents of the online help

The Qt library maps automatically the modifier key to the default of the operating system *neur*ALC is running on. As a result the command-key under MacOSX (used for illustration in the above figure) is reassigned to the Control-key (Ctrl) under Linux, or the Alternate-key (Alt) under Microsoft Windows.

7.2 Mouse Functionality


2D
click&drag
to mark rectangle, release to zoom into selected area of the visualization
click
actual values at the mouse position in the visualization plot are displayed
2D
click
to reset display view to full data view
3D
click&hold
to rotate 3D-view
3D
click&hold
to pan 3D-view
3D
use the scrollwheel to zoom 3D-view

Holding down the Shift-button while rotating the 3D-view restricts, depending on the mouse movement, the rotation to the y-axis. If you use a one-button mouse the 3D-view panning is possible by dragging the view while holding down the `Option`-key.

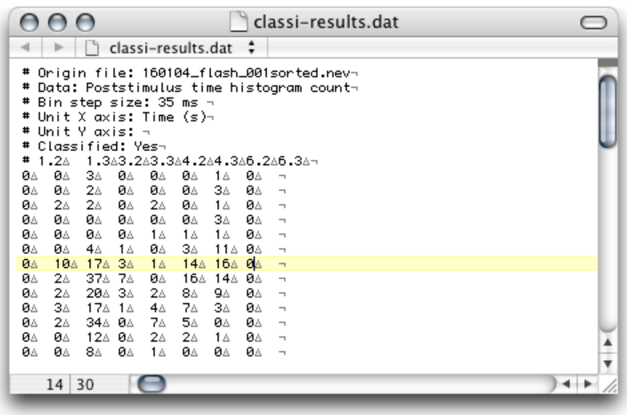
7.3 Supported File Formats

7.3.1 Input

*neur*ALC version 1.0.0 can only read neuronal multi-electrode recordings in NEV-files which comply with the NEV specification 2.0. The program checks the integrity of the data in time space when an experimental data file in this format is opened. Depending on the data acquisition it is possible that spike times are not filed in ascending temporal order. *neur*ALC validates and, if needed, aligns this order when opening a file.

7.3.2 Output

*neur*ALC version 1.0.0 supports a few output formats:
• All open neuronal data files can be saved in NEV2.x format, which allows to save TRC classification and modified comments to these files.
• Analysis results calculated for the selected electrode or neuronal “unit” data, can be exported and saved to an ASCII-file. Files written in ASCII-format by *neur*ALC have the following, general structure: a header of several lines, followed by columns which hold the analysis results (figure 39).



```
# Origin file: 160104_flash_001sorted.nev¬
# Data: Poststimulus time histogram count¬
# Bin step size: 35 ms ¬
# Unit X axis: Time (s)¬
# Unit Y axis: ¬
# Classified: Yes¬
# 1.2△  1.3△3.2△3.3△4.2△4.3△6.2△6.3△¬
0△  0△  3△  0△  0△  0△  1△  0△  ¬
0△  0△  2△  0△  0△  0△  3△  0△  ¬
0△  2△  2△  0△  2△  0△  1△  0△  ¬
0△  0△  0△  0△  0△  0△  3△  0△  ¬
0△  0△  0△  0△  1△  1△  1△  0△  ¬
0△  0△  4△  1△  0△  3△  11△ 0△  ¬
0△  10△ 17△ 3△  1△  14△ 16△ 0△  ¬
0△  2△  37△ 7△  0△  16△ 14△ 0△  ¬
0△  2△  20△ 3△  2△  8△  9△  0△  ¬
0△  3△  17△ 1△  4△  7△  3△  0△  ¬
0△  2△  34△ 0△  7△  5△  0△  0△  ¬
0△  0△  12△ 0△  2△  2△  1△  0△  ¬
0△  0△  8△  0△  1△  0△  0△  0△  ¬
```

Figure 39: Example ASCII-export for a selected group of neuronal "units"

Each header line begins with a #. This header holds information about the experimental data file used for the analysis results and the related calculation parameter. The last line of the header holds the "variables" or labels or the following, tabulator-separated result columns.

8 Bugs

Bugs? You really asking for known BUGS?!?? Geesh... well, we tried to make the program as bug-free, as possible. A description of the known "oddities" of *neur*ALC is provided within the *neur*ALC-distribution in the file `bugs.txt`. If you discover a bug, please refer for reporting to the following FAQ.

9 Frequently asked Questions

1. *"I'm using the Zimbutsu3000-XXL (autumn of 1984 edition) data-acquisition system (or whatever) which saves data in its own file format. neurALC in the current version seems not capable to open these files. When there will be support for this file format?"* - Well, this is GPL'ed sources - go ahead and extend the program for yourself. Of course we are open for suggestions, recommendations, etc. - feel free to email (but please do not expect immediate *well, if ever* implementation.)

2. *"I have found a bug. What should I do?"* - You can report all bugs (or possible fixes) through http://neuralc.sourceforge.net. Please include to the description the name of the operating system you are using, as well as the Qt-, QWT-, QWT3D-, and FFTW-versions.

3. *"After the installation of neurALC my previously installed version of the NEV2lkit (version 1.0-1.1.3) stop working. How to fix this??"* - this is a bug which is re-

lated to different version of the required libraries both programs are based on. Please visit the *NEV*2lkit home page at http://nev2lkit.sourceforge.net and download&install the latest version (⩾1.1.4) which addresses this problem.

4. *"I have successfully compiled neurALC and build an installer on the Sony's PSP, Qtopia, whatever else system and/or platform for which so far no binary installation is available. Do you like to include it in the neurALC distributions tree? "* - YES! Contact us by email or through http://neuralc.sourceforge.net

5. *"I have extended the import/output and/or analysis capabilities of neurALC. Do you like to include the changes to the neurALC source tree? "* - YES! Contact us by email or through http://neuralc.sourceforge.net

6. *"Well, Trolltech just now released the marvelous, wowed, effulgent, *please insert your descriptive expression here* Qt4 - including even a free desktop edition for Microsoft Windows. How can you dare to offer a free program based on the ancient, hoary, paleolithic, *please insert your descriptive expression here* Qt3?"* - We were and are aware of the release of Qt4 (and the resulting relief this has for example on the development of Qt-based programs under Microsoft Windows). *neur*ALC is partly based on extensions to the Qt3 framework, namely the QTW and QTW3D libraries. When we started the development neither of these libraries was reported to work fully without problems with Qt4. Well, and we had an educational license of Qt3, so... but, as mentioned previously, this is GPL'ed sources - go ahead and... (please see following entry in this FAQ)

7. *"I had some time during today's lunch break and ported the source code of neurALC to Trolltech's Qt4. Do you like to include the changes to the neurALC source tree? "* - HOW GREAT! THANKS!! And: YES! Please contact us by email or through http://neuralc.sourceforge.net

8. *"How to reference/quote/cite neurALC?"* - If you publish results in an article or use *neur*ALC as data processor for any other kind of publication, please use a phrase similar to: "Temporal response classes were obtained from the analyzed multi-electrode recordings using the *neur*ALC program" in the methods section, and include the following to your list of references: *neur*ALC - a cross-platform, open source program for the analysis of extra-cellular neuronal multi-electrode recordings distributed under GPL (http://neuralc.sourceforge.net).

10 Bibliography&References

M. Casdagli, Recurrence plots revisited, Physica D 108, 206 (1997)

G. Celeux and G. Govaert, A Classification EM algorithm for clustering and two stochastic versions, Computational Statistics and Data Analysis, 14(3):315-332. (1992)

J.P. Eckmann, S. Oliffson Kamphorst and D. Ruelle, Recurrence plots of dynamical systems, Europhys. Lett. 4, 973 (1987)

FFTW3 - library system for Fourier and related calculations. → `http://www.fftw.org`

A.M. Fraser and H.L. Swinney, Independent coordinates for strange attractors from mutual information, Phys. Rev. A 33, 1134 (1986)

KlustaKwik - a program for unsupervised classification of multi-dimensional continuous data (see KD Harris et al, Journal of Neurophysiology 84:401-414, 2000) → `http://klustakwik.sourceforge.net`

MySQL - an open source database. → `http://www.mysql.com`

NEV2lkit - a preprocessor for intra- and extra-cellular neuronal recordings distributed under GPL → `http://nev2lkit.sourceforge.net`

NEV-format specifications - as of this writing the Neural Event Format specifications (version 2.0) are publicly available. → `http://cyberkineticsinc.com/NEVspc20.pdf`

Qt - the cross-platform C++ framework from Trolltech. Please visit `http://www.trolltech.com`.

QWT - library for the Qt framework which provides Qt widgets for technical applications. → `http://qwt.sourceforge.net`

QWT3D - library for the Qt framework which provides Qt widgets for three-dimensional plotting. → `qwt3dplot.sourceforge.net`

TISEAN - a software project for the analysis of time series with methods based on the theory of nonlinear deterministic dynamical systems. → `http://www.mpiks-dresden.mpg.de/~tisean`

11 Licenses

As mentioned throughout this manual - *neur*ALC, the manual, examples and its sources are covered in full or partly by one of the following licenses. Copies of the GPL and LGPL can be found inside the *neur*ALC installation folder.

GNU Public License 2.0 (GPL)

The full text of the GPL is provided with *neur*ALC or can be found at
http://opensource.org/licenses/gpl-license.php

List of Figures

Contents

Colophon

This manual was written under MacOSX and layouted for two-sided printing on A4 (portrait) using TeXShop 1.40 and teTEX. The *neur*ALC and *db*ALC and some of the marginal icons were created by M. Bongard using Adobe Illustrator and Lemkesoft's GraphicConverter. Screen shots were taken under MacOSX 10.3.9 and 10.4.2 and processed using Apple's MacOSX preview application .