SigMate: A MATLAB–based automated tool for extracellular neuronal signal processing and analysis

Mufti Mahmud [a, c], Alessandra Bertoldo [b], Stefano Girardi [a], Marta Maschietto [a], Stefano Vassanelli [*, a]

[a] *NeuroChip Laboratory, Department of Human Anatomy and Physiology, University of Padova, via f. Marzolo 3, 35131 Padova, Italy*

[b] *Department of Information Engineering, University of Padova, via Gradenigo 6/B, 35131 Padova, Italy*

[c] *Institute of Information Technology, Jahangirnagar University, Savar, 1342 Dhaka, Bangladesh*

Abstract

Rapid advances in neuronal probe technology for multisite recording of brain activity have posed a significant challenge to neuroscientists for processing and analyzing the recorded signals. To be able to infer meaningful conclusions quickly and accurately from large datasets, automated and sophisticated signal processing and analysis tools are required. This paper presents a MATLAB–based novel tool, "SigMate", incorporating standard methods to analyze spikes and EEG signals and *in–house* solutions for local field potentials (LFPs) analysis. Available modules at present are – 1. *In–house* developed algorithms for: data display (2D and 3D), file operations (file splitting, file concatenation, and file column rearranging), baseline correction, slow stimulus artifact removal, noise characterization and signal quality assessment, current source density (CSD) analysis, latency estimation from LFPs and CSDs, determination of cortical layer activation order using LFPs and CSDs, single LFP clustering; 2. Existing modules: spike detection, sorting and spike train analysis, and EEG signal analysis. SigMate has the flexibility of analyzing multichannel signals as well as signals from multiple recording sources. The *in–house* developed tools for LFP analysis have been extensively tested with signals recorded using a standard extracellular recording electrode, and planar and implantable Multi Transistor Array (MTA) based neural probes. SigMate will be disseminated shortly to the neuroscience community under the open–source GNU–General Public License.

Key words: Neuronal signal analysis, neuronal signal processing, EOSFET, neuronal signal analysis software, brain–machine interfacing, local field potential.

* Corresponding author. Tel.: +39 049 8275337; fax: +39 049 8275301.

Email address: stefano.vassanelli@unipd.it (Stefano Vassanelli)

1. Introduction

Understanding neuronal networks functionality from brain signals recorded by means of neuronal probes require rigorous processing and analysis. Advances in microelectronics and microelectrodes technology have enabled scientists to record from hundreds of neurons and simultaneously from a number of channels (Buzsaki, 2004; Prochazka et al., 2001; Wise et al., 2004). Inferring meaningful conclusions by analyzing this massive amount of data recorded from noisy experimental conditions is a big challenge for the neuroscience and neuroengineering community (Buzsaki, 2004). Though individual tools are available to perform processing of EEG signals and neuronal spikes, yet no software is available which integrate signal processing and analysis of various types of signals including LFPs (Kwon et al., 2011; Quiroga et al., 2004).

There are a number of existing tools developed for academic and commercial purposes (Bokil et al., 2010; Bologna et al., 2010; Bonomini et al., 2005; Cui et al., 2008; Delorme and Makeig 2004; Egert et al., 2002; Goldberg et al., 2009; Gunay et al., 2009; Hazan et al., 2006; Herz et al., 2008; Huang et al., 2008; Kwon et al., 2011; Magri et al., 2009; Morup et al., 2007; Novellino et al.,2009; Quiroga et al., 2004; Smith and Mtetwa, 2007; Vargas-Irwin and Donoghue, 2007; Vato et al., 2004; Versace et al., 2008; Wagenaar et al., 2005). However, these tools mainly deal with data visualization, spike detection and sorting, spike train analysis, EEG signal analysis, and cross–software framework to include different software tools.

A couple of open platforms are under development to promote sharing of different laboratory–developed tools across the World Wide Web (Lidierth, 2009; Meier et al., 2008). Frequently, when neuronal signals are recorded using a software different from the one to be used for analysis, data format conversion is required. To address this issue, there exists an interface called “Neuroshare API” for standardizing file-formats and creating low-level handling and processing tools for neurophysiological experiment data (http://www.neuroshare.org/). The initiative is divided into two phases to obtain its goals: the first phase targets to create open library and format standards for the experimental data, and the second phase aims on developing free and open-source tools for low-level handling and processing of the data. The vendors provide Neuroshare libraries for their data formats to be used to read from their data files and analyze them. Though very useful, the Neuroshare's limitations are that not all platforms are supported, and data access is limited to the data items specified by the current standard. Continuous inclusion of vendor specific information to the proprietary data files makes them even harder to access.

Another initiative, the Neurophysiology Data Translation Format (NDF) developed at the CARMEN project (http://carman.org.uk/) aims to provide a means of sharing neurophysiology data (Fletcher et al., 2008; Watson et al., 2010). As per the specification (NDF, 2012), the NDF specifies a uniform file and format structure for data sharing and inter-application data communication using XML based configuration file to manage a variable amount of host data files. The NDF also specifies a set internal data types with the most commonly used experimental data entities applicable to signal or image processing applications, and include continuous time series, fixed/arbitrary length time series segments and spike times (e.g. events). NDF may be extended arbitrarily to accommodate other data entities.

Nevertheless, application of multiple packages on a single dataset to perform different operations becomes time consuming and cumbersome. An "umbrella" tool is required to perform these operations (Mahmud et al., 2010b). Table 1 provides a comparative study of available features among a few popular neuronal signal analysis tools.

Table 1: Comparison of available features in leading neuronal signal analysis tools

Tools	Features														
	DV	FO	Spike	LFP										EEG	
				NC	SAR	CSD	LC	CLAOD	LFPC	SA	CS	RDF	E/C L	SM	TFT
Chronux (Bokil et al., 2010)	Y	N	Y	N	N	N	N	N	N	Y	Y	Y	N	N	N
SPYCODE (Bologna et al., 2010)	Y	N	Y	N	N	N	N	N	N	Y	Y	N	N	N	N
DATA-MEAns (Bonomini et al., 2005)	Y	N	Y	N	N	N	N	N	N	N	N	N	N	N	N
BSMART (Cui et al., 2008)	Y	N	N	N	N	N	N	N	N	Y	Y	Y	N	N	N

EEGLAB (Delorme and Makeig 2004)	Y	N	N	N	N	N	N	N	N	N	N	N	Y	Y	Y
MEA-Tools (Egert et al., 2002)	Y	format conversion	Y	N	N	N	N	N	N	N	N	N	N	N	N
STAToolkit (Goldberg et al., 2009)	Y	N	Y	N	N	N	N	N	N	N	N	N	N	N	N
Klusters, NeuroScope, NDManager (Hazan et al., 2006)	Y	format conversion	Y	N	N	N	N	N	N	Y	Y	N	Y	N	N
NeuroQuest (Kwon et al., 2011)	Y	N	Y	N	N	N	N	N	N	N	N	N	N	N	N
SigMate (Proposed package)	Y	Y	Y (Adapted from Quiroga et al., 2004)	Y	Y	Y	Y	Y	Y	Y (Adapted from Bokil et al., 2010)			Y (Adapted from Delorme and Makeig 2004)		

Legends: DV (Data Visualization), FO (File Operations, particularly – file splitting, concatenation, column rearranging), Spike (includes Spike Detection, Sorting, and Train Analysis), NC (Noise Characterization), SAR (Stimulus Artifact Removal), LC (Latency Calculation), CLAOD (Cortical Layer Activation Order Detection), LFPC (LFP Classification), SA (Spectral Analysis), CS (Correlation Studies), RDF (Regression Data Fitting), E/C L (Event/Channel Localization), SM (Source Modeling), TFT (Time Frequency Transform). 'Y' denotes availability of a feature, and 'N' denotes absence of a feature.

In this paper we present a comprehensive software package, "SigMate", including our *in–house* algorithms to process and analyze LFPs along with standard packages for spike train and EEG signal analysis. Developed in MATLAB (version 7.9, R2009b, http://www.mathworks.com/) and tested in Windows 32 and 64–bit versions, this represents a versatile multipurpose package for processing and analyzing signals recorded using neuronal probes. A Graphical User Interface (GUI) environment facilitates the access to non-programming background users. The use case diagram of the software package can be seen in figure 1.

Due to the increasing usage of MATLAB in neuroinformatics research and selection of prototyping method for the development life-cycle, we decided to use MATLAB as a means of preliminary development platform for SigMate. However, future development plans include translation of SigMate as libraries for Python (http://www.python.org/) and platform independent executable files.

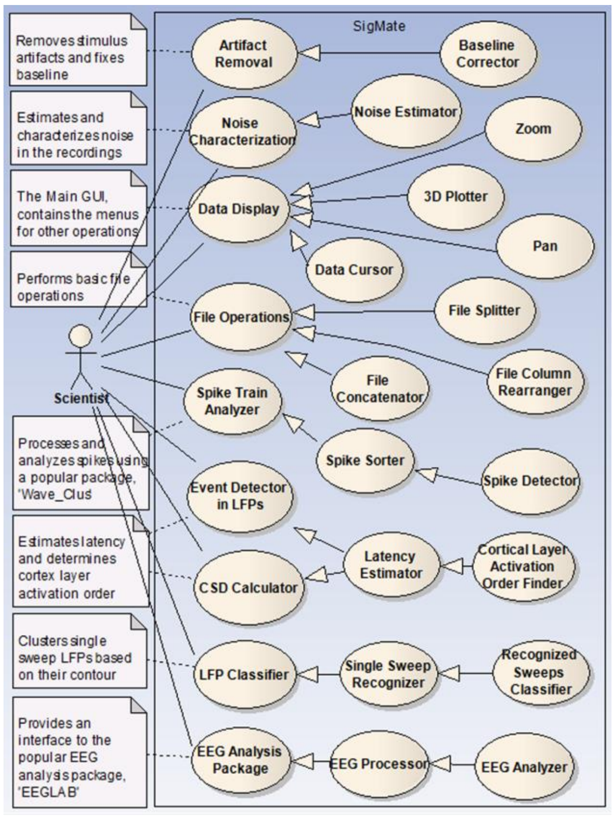

SigMate
Removes stimulus artifacts and fixes baseline
Estimates and characterizes noise in the recordings
The Main GUI, contains the menus for other operations
Performs basic file operations
Scientist
Processes and analyzes spikes using a popular package, 'Wave_Clus'
Estimates latency and determines cortex layer activation order
Clusters single sweep LFPs based on their contour
Provides an interface to the popular EEG analysis package, 'EEGLAB'
Artifact Removal
Baseline Corrector
Noise Characterization
Noise Estimator
Zoom
Data Display
3D Plotter
Pan
Data Cursor
File Operations
File Splitter
File Column Rearranger
File Concatenator
Spike Train Analyzer
Spike Sorter
Spike Detector
Event Detector in LFPs
Latency Estimator
Cortical Layer Activation Order Finder
CSD Calculator
LFP Classifier
Single Sweep Recognizer
Recognized Sweeps Classifier
EEG Analysis Package
EEG Processor
EEG Analyzer

Figure 1: Use case diagram of SigMate. This diagram provides an overview of the various modules present within the package.

2. Design and input–output

2.1. SigMate design

SigMate is designed through prototyping method using a multi–layered approach.

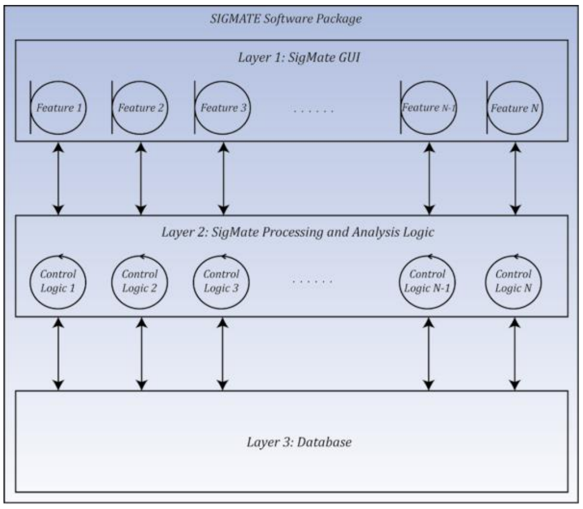


Figure 2: Three–layered architecture of the SigMate software package.

- *Presentation layer (top layer)*: This is the topmost level of the application. The presentation layer contains the GUIs of the application. It communicates with the middle layer by requesting the user commands.
- *Application layer (business logic, logic layer, or middle layer)*: The logic layer is separated from the presentation layer and here an application's functionality is controlled by performing detailed processing and analysis.
- *Data layer (bottom layer)*: This layer consists of databases and/or storages. Here information is stored and retrieved. This layer keeps data neutral and independent from applications or business logic. Giving data its own layer also improves scalability and performance. In SigMate, no databases are required, rather secondary storage devices are used to store the signals in files.

This type of three–layered architecture is a client-server architecture in which the user interfaces, functional process logic ("business logic"), computer data storage and/or data access are developed and maintained as independent modules. Apart from the usual advantages of modular software with well–defined interfaces, the three–layer architecture is intended to allow any of the three layers to be upgraded or replaced independently as requirements change. For example, a change of GUI design in the top layer would not affect the bottom two layers (Eckerson, 1995).

The figure 2 shows a diagram depicting the three-layered architecture of the SigMate software package. In the individual module descriptions (see Sec. 3) the communication diagrams are shown as a part of the method development.

2.2. Input–output and GUI handling

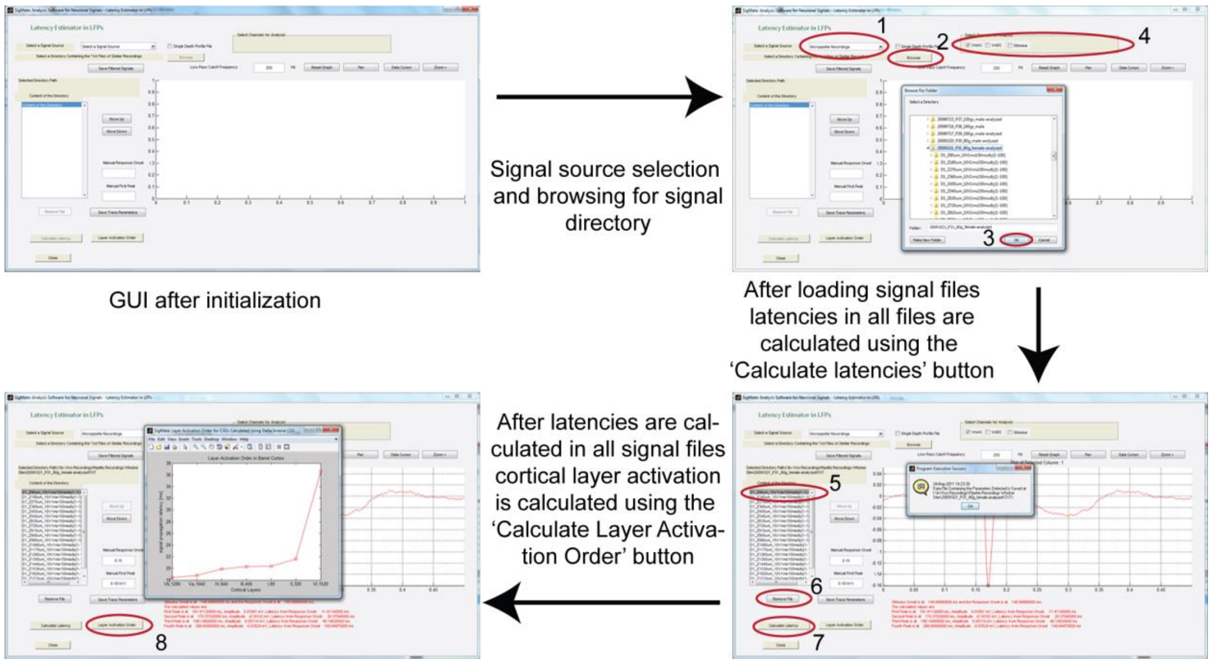
All the modules of SigMate are designed to accept ASCII text files containing data values separated by tabs. Although the ASCII format text files are heavier in size compared to many proprietary file types, we selected this file type due to its universal acceptance. Moreover, SigMate aims to provide a platform independent framework (not limited by the signal acquisition hardware) which can be achieved using the ASCII file type as most of the data acquisition systems allow their data to be saved, exported or converted to ASCII files.

The data values in a file should be stored in columns with the first column as the time vector and the other columns as recorded data values from different channels. For example, if a file contains recordings from a neuronal probe containing four recording sites or channels, the file is expected to have five columns: the first one is the recording times, and the other four are the recorded values by each site or channel. At the moment, SigMate modules can handle files containing simultaneous recordings from maximum five channels which will be extended to large numbers of channels in future releases.

Each GUI is having some common and useful features (as an example the data visualization GUI can be seen in figure 5B): 1) a dropdown box for selection of signal source, 2) a number of check boxes for channel selection, 3) a listbox to show the selected files, 4) a browse button to let the program select a folder whose contents can be seen in the listbox, 6) a display pane (with zoom, pan and data cursor add-ins) to visualize the selected signal file(s) from the listbox, 7) a remove file button to remove selective files from the listbox, 8) a load data button to load the signal files present in the listbox, and 9) module specific buttons and fields.

The selection of multiple signal files provides the flexibility for performing batch processing. At the moment it can handle signals from two possible signal acquisition sources: 1) single site neuronal probes (e.g., micropipettes, metal electrodes, etc.), and 2) multisite neuronal probes. In principle, during the processing and analysis, signals recorded using the single site neuronal probes can be treated as a special case of the multisite neuronal probes. However, frequently it is found that signals recorded using multisite neuronal probes are more prone to noise contamination compared to the conventional single site probes. For this reason, we treated signals coming from the two different sources differently.

Each module expects that the signals to be analyzed are stored in a folder which can be indicated to through the ‘browse’ button of a GUI. As an example, we illustrate below how a user can calculate the layer activation order from the recorded LFPs using the latency estimation module.



GUI after initialization
Signal source selection and browsing for signal directory
1
2
3
4
After loading signal files latencies in all files are calculated using the ‘Calculate latencies’ button
5
6
7
After latencies are calculated in all signal files cortical layer activation is calculated using the ‘Calculate Layer Activation Order’ button
8

Figure 3: An example of the GUI handling calculation of the latencies in LFPs to determine the cortical layer activation order (see section 3.5 for detailed information on the algorithm). The numbered marks (the red ovals) indicate the basic execution order of the program to be followed for performing the analysis. 1. Signal source selection (from the signal source dropdown list. In this case, ‘micropipette recordings’ was selected), 2. Browse for the signal directory (a directory containing the signal files. In this case, there were 20 files each one corresponding to a recording position and containing two columns: first one is the recording time and the second one is the recorded data), 3. Click ok in the browse window, 4. Channel selection (using the ‘select channel’ checkboxes, one channel was selected as micropipettes provide single channel recordings), 5. Clicking on each recording in the listbox displays the signal in the display pane, 6. Files can be removed from the listbox using the ‘remove file’ button, thus, excluding them from analysis, 7. Click the ‘Calculate Latency’ button which calls a function capable of detecting events present in each of the files and of calculating and saving their latencies, and 8. The cortical layer activation

order is determined using the ‘Calculate Layer Activation Order’ button and displayed in a new window.

3. Methods

The software package is designed to perform various processing and analysis on the neuronal signal files. The functionalities of the software includes adoption of existing popular tools for spike train analysis (‘Wave_Clus’ by Quiroga et al., 2004) and EEG signal analysis (‘EEGLAB’ by Makeig, 2004), in-house developed tools for processing and analysis of LFPs, and three more LFP analysis features (spectral analysis, correlation studies, and regression data fitting) from Bokil et al., 2010. The in-house developed algorithms for file operations and LFP analysis are described in detail in the subsections 3.1 to 3.6. The non-exhaustive list (due to continuous inclusion of new features) of features present at the moment are: data display / visualization (2D and 3D) with zooming, panning and data cursor options, slow stimulus artifact removal (including baseline correction), noise characterization and signal quality assessment with noise estimation, file operations (including file splitting, file concatenating and file column rearranging), latency estimation (in LFPs and CSDs) with the possibility to detect the cortex layer activation order, and single LFP sorting. These features were extensively tested with event related neuronal signals recorded using conventional single site probes (standard borosilicate micropipette), and Electrolyte-Oxide-Semiconductor Field Effect Transistor (EOSFET) based multisite neuronal probes from anesthetized rats (see supplementary document for details on signal acquisition techniques and Felderer and Fromherz 2011; Hierlemann et al., 2011; Hutzler et al., 2006; Lambacher et al., 2011; Vassanelli and Fromherz 1997, 1998, 1999; and Vassanelli et al., 2012 for examples of signals acquired by transistor based probes). However, the stimulus artifact removal, and baseline correction may be used with non-triggered signals. The following subsections describe the *in-house* developed LFP analysis algorithms (a conceptual schematic diagram can be seen in figure 4) of the individual modules in greater details.

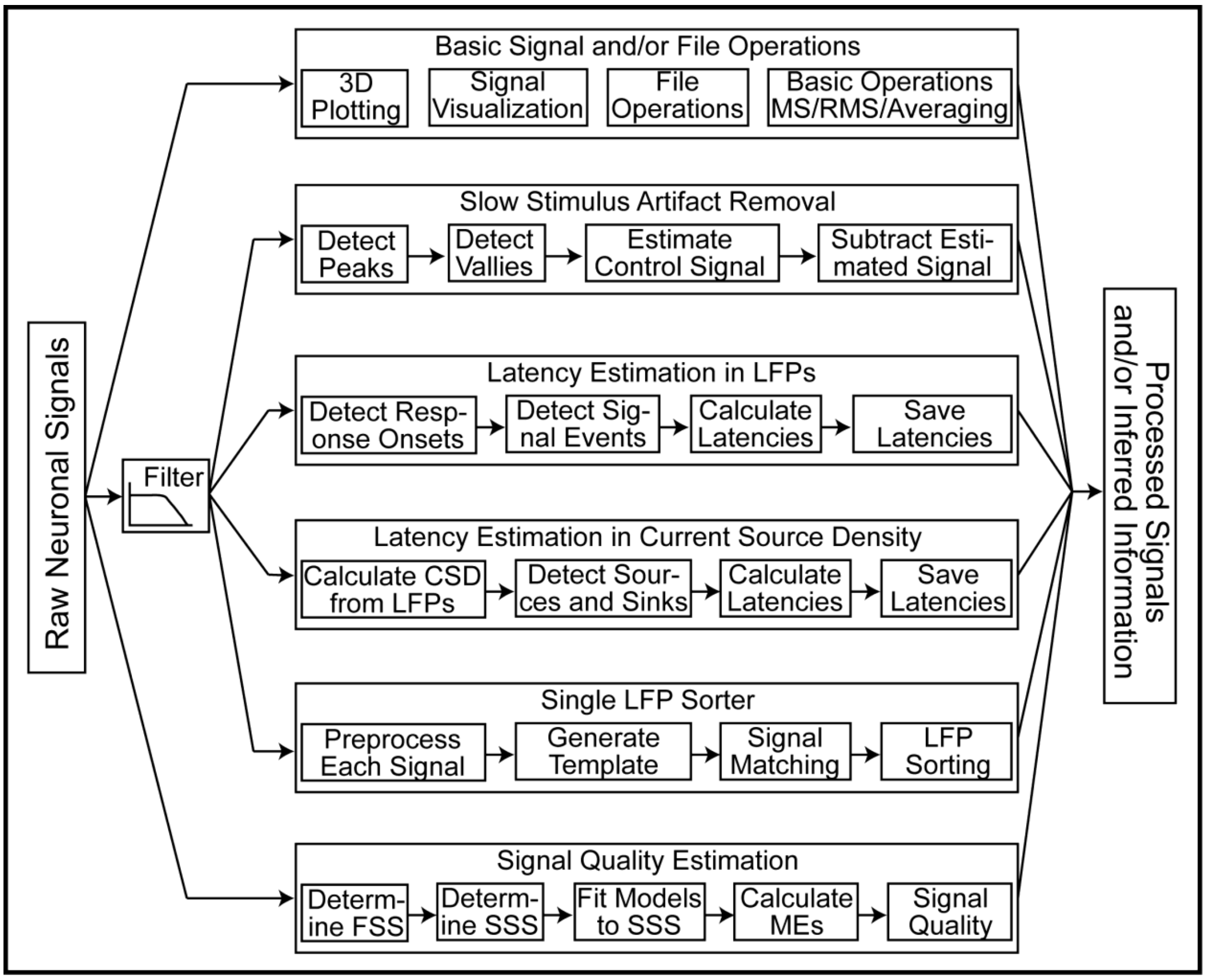


Figure 4: Conceptual schematic diagram of the in-house algorithms outlining major steps for each operation.

3.1. Data display

This is the initial GUI, i.e., the application is launched using this GUI. It contains the menu bar incorporating links to other modules of the SigMate. Its functionality does not only provide the user with the flexibility in viewing the signals in 2D and 3D, but also provide the possibility to perform averaging of single sweep signals, estimate the noise, perform ± averaging, and calculate the mean square (MS) and root mean square (RMS) of the signals. Noise estimation through RMS and MS is used to gain preliminary noise information on the signals. A dedicated module for noise characterization and signal quality assessment is also included in the package (see sec. 3.4). The communication diagram and the GUI can be seen in figure 5.

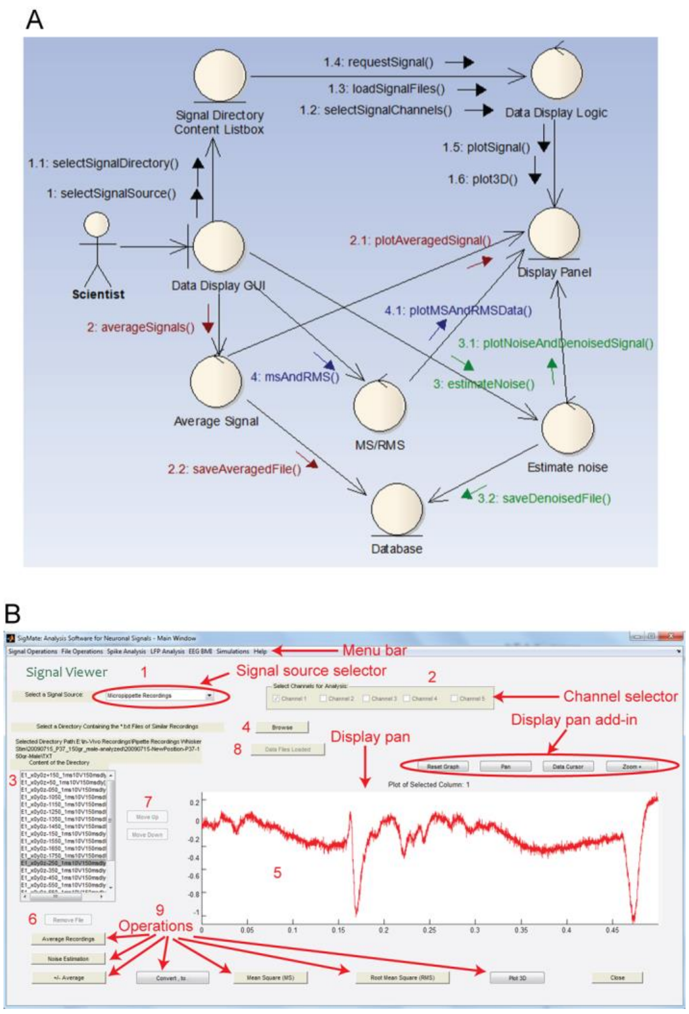


Figure 5: A: Communication diagram of the data display module. B: GUI of the data display module and its possible operations. The menu bar contains menus that are linked to the other modules of the package.

3.2. File operations

Basic file operations are being incorporated in the software package that are often time consuming for the scientists who use different software for signal recording and performing signal processing and analysis. As different tools require signal files to be formatted in their particular ways, this module performs three basic and important formatting operations: file splitting (splits a multi–sweep file into single–sweeps based on the sampling frequency), file concatenation (concatenates multiple single–sweep files into a multi–sweep file), and file column rearranging (retains only the selected channels and eliminates the unselected ones). Figure 6 shows the communication diagram of the module.

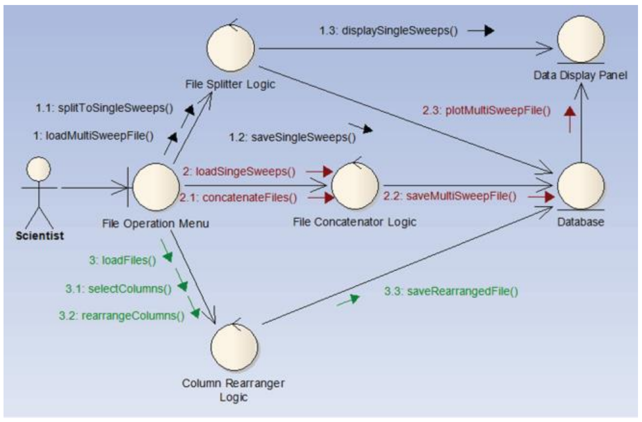


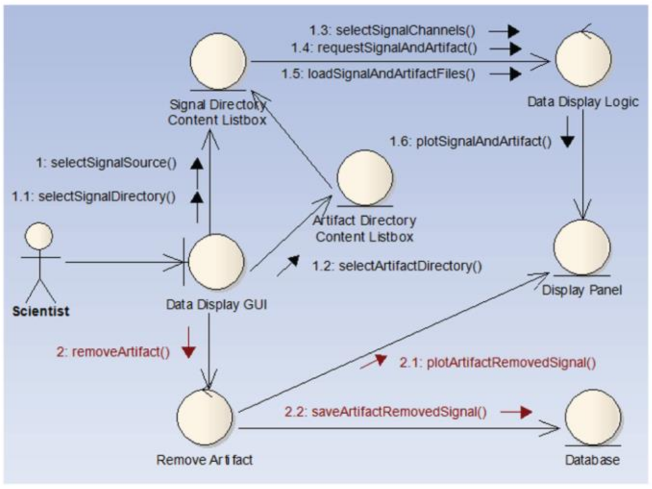
Figure 6: Communication diagram of the file operation module.

3.3. Slow stimulus artifact removal

In general, neuronal signals can be contaminated with two different types of stimulus artifacts: slow and fast. Usually, frequency components of the fast artifacts are in range of 1–3 kHz, whereas slow artifacts lay below 100 Hz overlapping with frequency components of typical LFPs. The slow stimulus artifact removal module performs slow artifacts removal and baseline correction of the cortical surface recorded signals. In our recordings, a slow artifact was caused by the bath modulation during application of air–puff stimulation and laid within the range of 8–12 Hz. As the frequency of the artifact overlapped with the response, the method used control signals (recorded after the brain activity was suppressed) to remove the artifacts from the signals with evoked responses.

The removal is performed using an *in–house* algorithm (listed in appendix A.1). This algorithm detects the peaks and valleys in a signal (S_{org}). For each peak there is a corresponding valley which constitutes a peak–valley pair for a tiny signal part. Once all peak–valley pairs in a signal are detected, their average provides an estimation of the signal (S_{est}). The mean of this estimated signal ($\mu_{s\text{-}est}$) is subtracted from the original signal for baseline correction ($S_{org} - \mu_{s\text{-}est}$). The estimated signal ($S_{est\text{-}ctrl}$) calculated using the control signal (S_{ctrl}) is subtracted from the evoked signal to remove artifact from the evoked signal, i.e., $S_{artifact\text{-}removed} = S_{org} - {}_{Sest\text{-}ctrl}$ (Mahmud et al., 2009a). Figure 7 shows the communication diagram and the GUI of the module and figure 8 shows flowchart of the artifact removal method.

A.


1.3: selectSignalChannels()
1.4: requestSignalAndArtifact()
1.5: loadSignalAndArtifactFiles()
Signal Directory Content Listbox
Data Display Logic
1.6: plotSignalAndArtifact()
1: selectSignalSource()
1.1: selectSignalDirectory()
Artifact Directory Content Listbox
1.2: selectArtifactDirectory()
Scientist
Data Display GUI
Display Panel
2: removeArtifact()
2.1: plotArtifactRemovedSignal()
2.2: saveArtifactRemovedSignal()
Remove Artifact
Database

B.

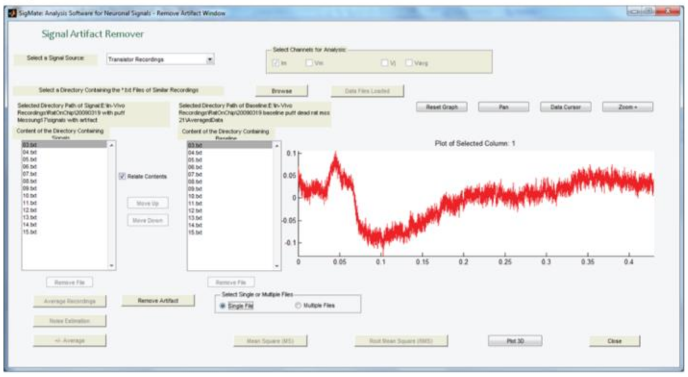

SigMate: Analysis Software for Neuronal Signals - Remove Artifact Window
Signal Artifact Remover
Select a Signal Source:
Transistor Recordings
Select Channels for Analysis:
Im
Vm
Vj
Vavg
Select a Directory Containing the *.txt Files of Similar Recordings
Browse
Data Files Loaded
Selected Directory Path of Signal:E:\In-Vivo Recordings\RatOnChip\20090319 with puff Messung17\signals with artifact
Selected Directory Path of Baseline:E:\In-Vivo Recordings\RatOnChip\20090319 baseline puff dead rat mess 21\AveragedData
Content of the Directory Containing Signals
Content of the Directory Containing Baseline
Reset Graph
Pan
Data Cursor
Zoom +
Plot of Selected Column: 1
Relate Contents
Move Up
Move Down
Remove File
Average Recordings
Remove Artifact
Noise Estimation
+/- Average
Select Single or Multiple Files
Single File
Multiple Files
Mean Square (MS)
Root Mean Square (RMS)
Plot 3D
Close

Figure 7: A: Communication diagram of the artifact removal module. B: GUI of the artifact removal module offering the possibility to perform artifact removal on single signal files or batch processing of multiple files.

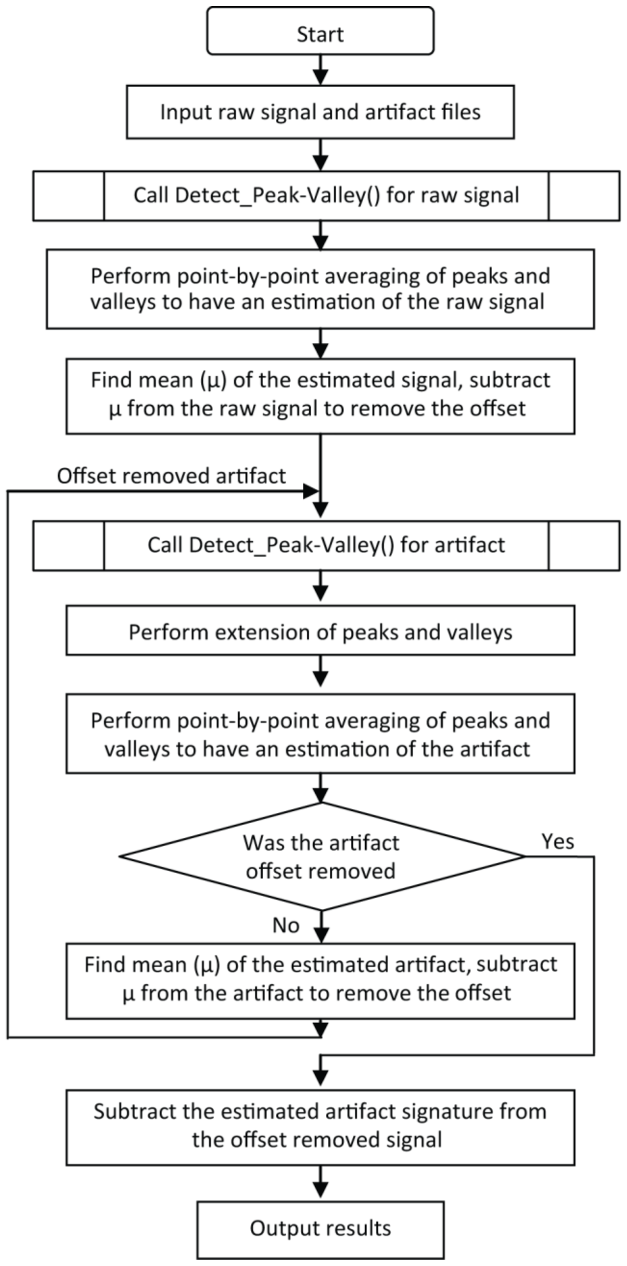

Start
Input raw signal and artifact files
Call Detect_Peak-Valley() for raw signal
Perform point-by-point averaging of peaks and valleys to have an estimation of the raw signal
Find mean (μ) of the estimated signal, subtract μ from the raw signal to remove the offset
Offset removed artifact
Call Detect_Peak-Valley() for artifact
Perform extension of peaks and valleys
Perform point-by-point averaging of peaks and valleys to have an estimation of the artifact
Was the artifact offset removed
Yes
No
Find mean (μ) of the estimated artifact, subtract μ from the artifact to remove the offset
Subtract the estimated artifact signature from the offset removed signal
Output results

Figure 8: Flowchart of the artifact removal method.

3.4. Noise characterization and signal quality assessment

The noise characterization module is designed to be able to assess the quality of the recorded signals and quantify the noise. It uses *in–house* algorithms to detect the first steady–state (FSS, the pre–response portion of the signal), and the second steady–state (SSS, the post–response portion of the signal). After fitting polynomial models to the detected steady–states, it calculates the measurement error (ME) present in the signal (Mahmud et al., 2009b; 2011d). Statistical information (mean and standard deviation, SD) and distribution of the ME are used in quantifying the noise, thus providing an assessment of the signal quality (Bowman and Azzalini, 1997). The algorithms of ME calculation, FSS and SSS detection are listed in appendices A.2, A.2.1 and A.2.2, respectively.

3.4.1. Signal quality assessment using measurement errors (MEs)

The signal quality assessment starts with the hypothesis that the noise present in the signal is of Gaussian distribution with zero mean. Therefore, mathematically the noisy part of the signal can be represented by the equation 1 (Dodge, 2003).

$$z(t) = y(t, \mathbf{p}) + e(t) \tag{1}$$

where *z(t)* are the measured values, *e(t)* are the noise, and *y* is the mathematical representation of the true values. Here, *y* is a function of time (*t*) and depends on the parameter vector $\mathbf{p} = [p_1, \ldots, p_n, p_{n+1}]^T$. In this kind of representation, time is the only independent variable and the measurements are made precisely at known times, t_i, $i = 1, \ldots, N$.

The steady–state portions of the signal are represented by an n–order polynomial model whose coefficients (in form of parameter vector, **p**) are calculated using weighted least square using QR decomposition of the Vandermonde matrix constructed from *t* and stored in descending order of powers.

Weighted residual sum of squares (WRSS) is used in estimating the *y(t,* **p***)*. The parameter estimation is obtained by minimizing the weighted sum of squared difference between the observed *z(t)* and the model predictions *y(t,* **p***)* for a number of candidate models (*j=1,…,m*). The weights (*w*) are calculated using the variance σ_{ji}^2, with $w_{ji} = 1/\sigma_{ji}^2$ (Landlaw and DiStefano, 1984).

However, for each of the *m* candidate models of order *n*, the parameter vector **p**, the predicted data *y(t,* **p***)*, and the WRSS are calculated. Then, to select the optimal model in terms of polynomial order (that best fits the data) from a set of possible candidate models, the Akaike information criterion (*AIC*) is applied (Akaike,1974).

The minimal *AIC* value represents the optimal model with order *n* and parameter **p**, and implies that the model with this set of parameters will provide best results in calculating the ME using equation 1.

3.5. Latency estimation

There are two different modules for the latency estimation. The first one calculates latencies in LFPs by detecting various signal events. The other one first derives the CSDs from the LFPs and then detects the first sinks' peaks for the latency calculation. For both the LFPs and CSDs, the latencies are calculated as differences between the stimulus–onset and the respective events.

Neuroscientists often rely on barrel cortex LFPs to investigate the somatosensory system of some rodents (Ahissar and Knutsen, 2008; Diamond et al., 2008). For cortical signals, latencies provide information about the propagation time required for a stimulus evoked event to reach a cortical position. We used the latencies to determine the layer activation order from LFPs and CSDs. The estimation of latencies requires detection of events to be treated as reference points. For the cortical LFPs, the layer specific characteristics or events are explored using an *in-house* signal derivative based algorithm (Mahmud et al., 2011a). The latency of an event is calculated as the difference between the event's occurrence time and the stimulus-onset. The layer activation order is calculated using the latencies of the most prominent events across different cortical depths. For the CSDs calculated from the LFPs, we consider the first occurring sink as the reference event and calculate the latencies as the difference between the first sink's peak and stimulus-onset. For both LFPs and CSDs, the calculated latencies are associated with recording depths known *a-priori* and grouped into layers based on the depth information. The minimum latency from each layer is found and sorted in ascending order to obtain the layer activation order. The method is discussed in detail in the following subsections.

3.5.1. Barrel cortex LFPs

The LFPs recorded from a barrel column of the rat S1 cortex by stimulating the corresponding whisker can be differentiated by their specific characteristics based on the depth or layer they are recorded from. It's worth mentioning here that, barrel cortex LFPs can be represented using a template of four subsequent events (E1–E4) (Ahrens and Kleinfeld, 2004; Alloway, 2008; Kublik, 2004; and Mahmud et al., 2011a). These events are automatically detected and used in automated estimation of latencies. A representative signal depth profile recorded from the rat barrel cortex can be found in the accompanying supplementary document.

3.5.2. Event detection and latency estimation from LFPs

The event detection for each barrel cortex LFP is performed automatically using an *in–house* algorithm (Mahmud et al., 2010a). The algorithm is based on derivative calculation and detects in each file: the stimulus–onset, the response–onset, and the four events representing the barrel cortex LFPs (E1, E2, E3, and E4). Once the events are detected, latencies are calculated by subtracting the occurrence time of the events from the stimulus–onset time and are saved in a file for further processing (Mahmud et al., 2011a).

3.5.3. CSD calculation and latency estimation in CSDs

Due to the widespread use of CSD analysis in the neuroscience community, we have included a module to calculate the CSD profile from the recorded LFP profile (Armstrong–James et al., 1992; Castro–Alamancos and Oldford, 2002; Di et al., 1990; Einevoll et al., 2007; Jellema et al., 2004; Kaur et al., 2004; Megevand et al., 2009; Mitzdorf and Singer, 1980; Mitzdorf, 1985; Rappelsberger et al., 1981; Staba et al., 2004; Swadlow et al., 2002; Szymanski et al., 2009). The CSD computation is adapted from Pettersen et al., 2006 to compute CSD using four different methods (standard, delta–inverse, step–inverse, and spline–inverse CSD methods). The δ-Source Inverse CSD method (δ-Source iCSD) and its application to barrel cortex LFPs are explained in detail in a previous publication (Mahmud et al., 2011a).

After the CSD profile is computed, the sources and sinks for the individual recording site can easily be viewed. The calculation of the sinks' latencies is done by subtracting the time instance of the stimulus–onset from the time instance of the peak of the first sink.

3.5.4. Layer activation order detection

An important feature of the latency estimation module is that it can automatically calculate the cortical layer activation order based on the estimated latencies from the LFPs and the CSDs. Once the latencies are determined from the LFP profile and its respective CSD profile, they are lalyerwise grouped based on *a-priori* information regarding signal's recording positions. The latency of E2 is used in calculating the layer activation order from the LFPs. The ascending sequence of the layerwise minimum latencies provided the layer activation order.

3.6. Clustering of single LFP signals

The LFPs provide a finger–print of the stimuli's effect on signal generation and propagation of neuronal networks in the brain region under study (Legatt et al., 1980). Conventionally, these LFPs are recorded for a period of time and then a stimulus–locked average is obtained for analysis. However, previous studies show that averaging single LFPs causes information loss in the averaged LFP (Van Hemmen and Ritz, 1995). Also, to investigate certain brain functional properties (for example, signal processing pathways) and for certain operations (for example, current source density analysis) signal shape plays an important role (Okun et al., 2010; Mahmud et al., 2011a). As different shapes in the single LFP signals denote different neuronal network activities, a shape based classification method is necessary.

The method performs the clustering in three major steps: (i) smoothing / estimation of single LFPs and template generation, (ii) single LFP recognition, and (iii) classification of recognized single LFPs.

The single LFPs contain spontaneous brain activity along with the stimulus evoked response. To remove the oscillatory spontaneous brain activity, smoothing / estimation is performed using nonlinear least square estimation. The main reasons behind performing the estimation are two folds. Firstly, reduction of noise without distorting the shape information often caused by filtering, and secondly, facilitating the recognition of signal characteristics to be used as the base for selecting the feature vector for the clustering algorithm. Once the signals are estimated, the starting and end of the response is determined (response part) by truncating the pre-stimulus and post-response (the part after response till the end of the signal) parts. An average of these response subsets is usually considered as a template for single LFP recognition. This method makes use of contour matching for recognition of the single LFPs. The contour of the generated template is compared to each of the single LFP's contour with a predefined boundary condition. The single sweeps that fall within the boundary condition are considered to be recognized. Once the single LFPs are recognized, intelligent K–means clustering is applied on the recognized signals to classify them according to their shapes. The classified or clustered single sweeps are then locally averaged and saved for further processing (Mahmud et al., 2010c; 2011c).

3.6.1. Signal smoothing / estimation and template generation

Precise information about the signal events are often obscured by spontaneous neural oscillations and noise present in the single LFPs. To get rid of these oscillations and noise, nonlinear least square method is used for smoothing / estimating the single LFPs.

Using least square, for a given vector function $f\colon \Re^{n} \rightarrow \Re^{m}$ with $m \geq n$, we want to minimize ||$f(x)$|| or equivalently find:

$$x^{*} = argmin_{x}\{F(x)\} \qquad (2)$$

Where x^* is the local minimizer of $F(x)$. For a set of arguments, x^*, the value of $F(x)$ is kept minimal within the range of a very small positive integer, δ (Madsen et al., 2004). The $F(x)$ can be calculated using:

$$F\left(x\right)=\frac{1}{2}\left(\sum_{i=1}^{m}(f_i(x))^2\right)=\frac{1}{2}\left(\| f(x)\|^2\right)=\frac{1}{2}\left(f\left(x\right)^T f\left(x\right)\right) \tag{3}$$

Now considering a model to calculate the prediction error (ν) (equation 4) and adding the covariance matrix of this prediction error (Σ_ν) as a weight function to equation 2 and 3, an analytical solution of the problem can be obtained (equation 5).

$$x=y\left(x^*\right)+\nu \tag{4}$$

$$x^*=\left(y^T\Sigma_\nu^{-1}y\right)^{-1}y^T\Sigma_\nu^{-1}x \tag{5}$$

The nonlinear problem then can be solved using first order Taylor's expansion around the initial value of the parameter vector ($x_k^*, k=0$):

$$\Delta x=P\Delta x^*+\nu \tag{6}$$

Here, *P* is the partial derivative matrix with the predicted values using the initial set of parameters. A linear formula now can be used to estimate these parameters (equation 7) and to calculate the new parameter vector (equation 8). This iterative process is repeated until the cost function stabilizes or falls below a threshold.

$$\Delta x^*=\left(P^T\Sigma_\nu^{-1}\right)^{-1}P^T\Sigma_\nu^{-1}x \tag{7}$$

$$x_{k+1}^*=x_k^*+\Delta x_k^* \tag{8}$$

The estimated signals are scanned for occurrences of the events (E1–E4). From each signal the part from the stimulus–onset till the E4 are extracted for the calculation of the template. These extracted parts are usually different in lengths, thus, to obtain parts with same length, the longest part is selected and the rest are zero–padded. The average of all these parts provides the template.

3.6.2. Single sweep recognition

After the template's generation, boundary conditions are imposed on it and its contour is used to recognize the single LFPs. The upper and lower bounds are calculated using equations 9 and 10.

$$Up(k) = Temp(k) + (a \times (V_{tmp}(k))^{1/2} + b) \tag{9}$$

$$Low(k) = Temp(k) - (a \times (V_{tmp}(k))^{1/2} + b) \tag{10}$$

with *a, b* are constants; the values of *a, b* are determined empirically (*a = σ(Temp)*, and *b =3× σ(Temp)*), and V_{tmp} is the template's variance vector (equation 9).

$$V_{tmp} = \frac{1}{N} \sum_{i=1}^{N} [Sw_i(k) - Temp(k)]^2 \tag{11}$$

where *Sw* is the zero–padded and truncated single sweeps and *Temp* is the template.

A signal is considered to be recognized, if and only if all of its data points lie within the range of the boundary conditions.

3.6.3. Clustering the recognized sweeps

For our purpose we used the intelligent K–means (*i*K–Means) method of classifying recognized sweeps (Chiang and Mirkin, 2010). It is an updated version of the classical K–means (Bock, 2007; Macqueen, 1967). In the rest of the text the words classification and clustering are used synonymously.

The K–means method usually is applied to a dataset involving a set of *N* entities, *I*, a set of *M* features, *V*, and an entity–to–feature matrix $Y=(y_{iv})$, where y_{iv} is the value of feature $v \in V$ at entity $i \in I$. The method produces a partition $S = \{S_1, S_2, \ldots, S_K\}$ of *I*, an *M* dimensional vector in the feature space $(k=1, 2, \cdots, K)$, in *K* non–overlapping classes S_k (referred to as clusters) with centroids $c_k=(c_{kv})$. The centroids form a set $C=\{c_1, c_2, \ldots, c_K\}$ and the clusters are decided basing on a minimization criterion of within–cluster distance to centroids (equation 12).

$$W(S,C) = \sum_{k=1}^{K} \sum_{i \in S_k} d(i, c_k) \tag{12}$$

with *d* is the square of calculated Euclidean distance.

Given *K M*–dimensional vectors with c_k as cluster centroids, the algorithm updates clusters S_k according to the Minimum distance rule: for each entity *i* in the data table, its distances to all centroids are calculated and the entity is assigned to its nearest centroid. Given clusters S_k, centroids c_k are updated according to the distance d in equation 12. Specifically, c_k is calculated as the vector of within–cluster averages as d in equation 12 is Euclidean distance squared. This process is reiterated until clusters S_k stabilize.

The method uses an anomalous pattern (AP) based algorithm to find out the appropriate number of clusters at the beginning of the clustering (Chiang and Mirkin, 2010). The AP algorithm starts with an entity as the initial centroid c which is the farthest from the origin. After that, a one–cluster version of the generic K–Means is utilized. The current AP cluster S is defined as the set of all those entities that are closer to c than to the origin, and the next centroid c is defined as the center of gravity of S. This process is iterated until convergence.

Finally, when the single sweeps are classified into their respective clusters, they are cluster–wise averaged for further processing.

4. Results and discussion

The features of the package were tested with datasets recorded using three different recording methods. 1) Using standard borosilicate micropipettes as extracellular single–site electrodes; 2) using EOSFET based planar multi transistor array chips; and 3) using EOSFET based implantable chips. The supplementary document contains detailed description about the setup and the recording methods. As the spike and EEG analysis modules were adapted from popular and widely tested tools, we just provided interfaces between SigMate and those tools. Each of the *in–house* modules contain a GUI and they all are kept user friendly so that the operations can be performed easily even by non–programming background users rather than typing commands in the command line. The following subsections demonstrate the features workability on representative datasets.

4.1. Artifact removal

The figure 9 shows the estimation of a control signal calculated by the peak–valley detection algorithm described in the appendix A.1. The offset of this control signal was corrected using the mean of its estimation.

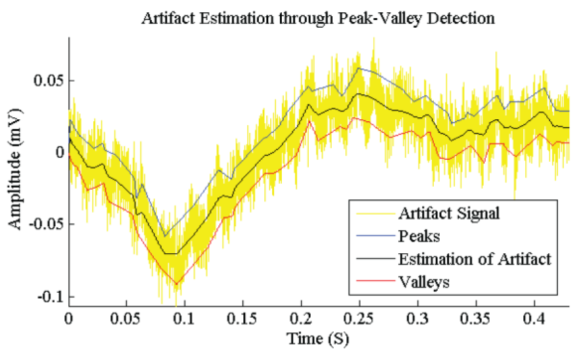


Figure 9: Control signal and its estimation through peak–valley detection using the signal‟s standard deviation as threshold.

Figure 10 shows the traces before and after stimulus artifact removal. The top trace is the artifact, the middle trace shows the evoked signal contaminated by artifact, and the bottom trace is the signal after artifact removal. The air–puff stimulation is shown at the bottom.

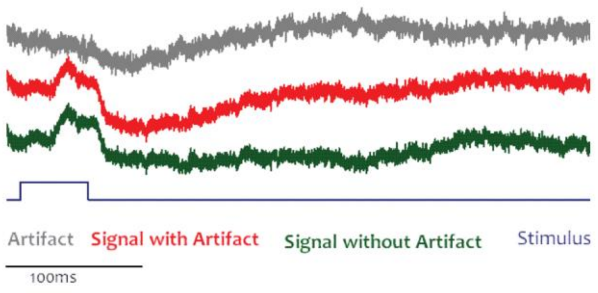


Figure 10: Traces of control signal (gray), evoked potential with artifact (red), artifact removed evoked potential (green), and the stimulus.

This method was also applied on a number of signals to perform batch processing. The figure 11(a) shows a 3D plot of simultaneously recorded signals from 13 EOSFETs with stimulus artifact (the two arrows point the artifact region). The figure 11(b) shows the 3D plot of the same signals after artifact removal by batch processing of the artifact removal module.

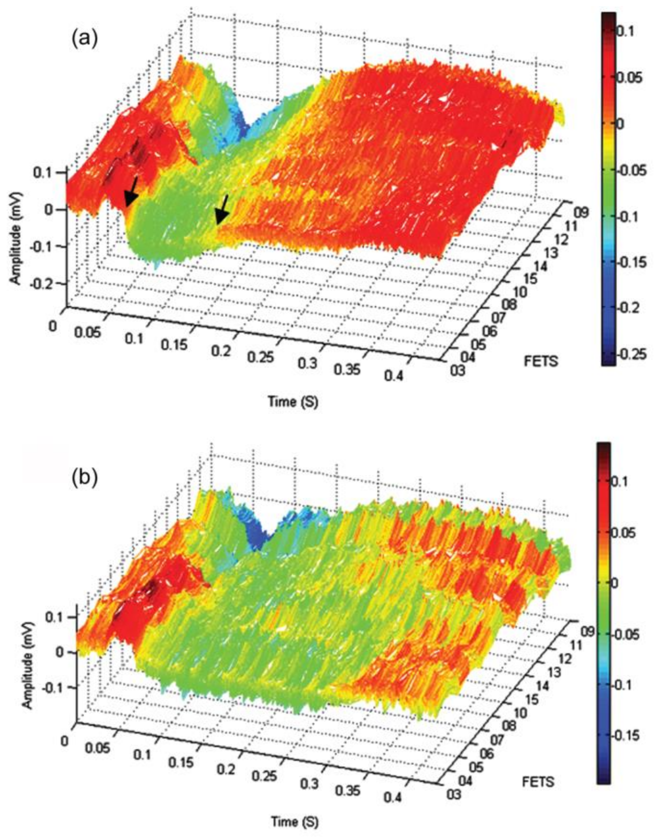


Figure 11: 3D signals before and after stimulus artifact removal. The color–bars show the amplitude intensity of the signals. (a) Raw signals recorded from 13 EOSFETs. The two arrows show the stimulus artifact region. (b) Signals without stimulus artifact as a result of batch processing.

4.2. Noise characterization and signal quality assessment

The method was tested on different datasets for a range of candidate models (m = 6) and order (n = 2 to 7). Except for a few signals it could calculate the steady–states accurately and provide successful signal quality assessment. In case of highly oscillatory signals with a high SD, the method failed to calculate the accurate response–onset, thus, the steady–states. In a pool of 65 different datasets, the algorithm failed to detect the exact evoked response for 5% of the signals. During an experiment, the multisite neuronal probes acquire signals simultaneously with an assumption that the experimental conditions for all the sites are similar. Considering this hypothesis, the quality of the recordings from an experiment can be assessed neglecting the failed cases.

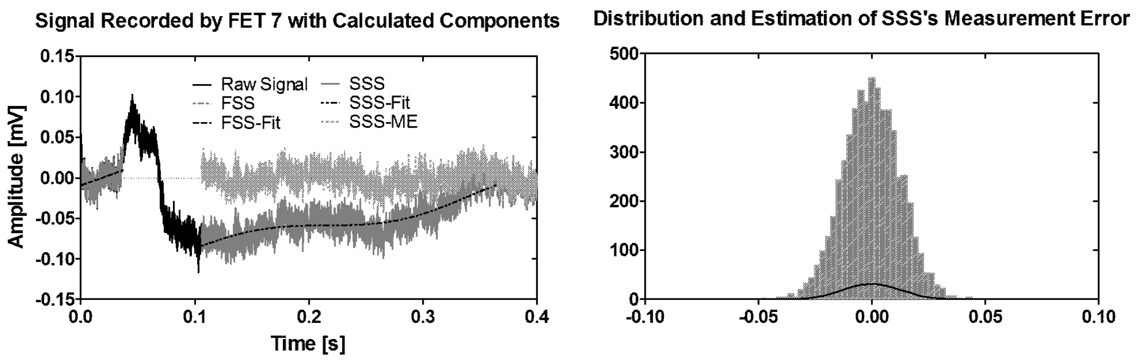


Figure 12: Left: figure showing the raw trace (black solid line); the detected FSS (grey dot-dash line); the SSS (grey solid line); fitted polynomial model, SSS–Fit (black dot-dot-dash line); and the SSS"s ME, SSS–ME (dashed grey line). Right: histogram of statistical distribution of SSS"s ME and its estimated density function (Gaussian).

Figure 12 shows representative result of FSS and SSS's detection (left). It depicts the raw trace (in black solid line) with detected FSS (gray dot–dash line), the FSS fit to facilitate the detection of the response–onset (dashed black line), the SSS (gray solid line), the polynomial model fitted to the SSS (black dot–dot–dash line), and the ME of the SSS (dashed gray line). The fitted polynomial model is seen as an oscillating wave as the post–response portion of the signal usually contained spontaneous oscillatory brain activity uncorrelated to the given stimuli. The ME calculated from the SSS had a Gaussian distribution as hypothesized.

Table 2 reports the means and SDs of the FSSs and the SSSs and their respective MEs. The steady−states were calculated using the algorithm listed in appendix A.2. Analyzing these results we noticed that the means of the MEs for all the signals (μ_{me-fss} and μ_{me-sss}) were very close to zero and the SDs of these MEs (σ_{me-fss} and σ_{me-sss}) were consistent around 0.01.

Table 2: Mean and SD of FSS, SSS with their MEs

FET	μ_{fss}	σ_{fss}	$\mu_{me\text{-}fss}$	$\sigma_{me\text{-}fss}$	μ_{sss}	σ_{sss}	$\mu_{me\text{-}sss}$	$\sigma_{me\text{-}sss}$
FET01	0.0067	0.0123	0.0831×10^{-3}	0.0118	0.0016	0.0153	9.397×10^{-15}	0.0108
FET02	0.0038	0.0114	0.0618×10^{-3}	0.0109	0.0038	0.0118	1.091×10^{-12}	0.0110
FET03	0.0013	0.0105	0.0750×10^{-3}	0.0101	0.0068	0.0101	1.513×10^{-13}	0.0096

FET04	0.0098	0.0144	0.0036×10^{-3}	0.0139	0.0014	0.0144	1.612×10^{-14}	0.0119
FET05	0.0044	0.0133	0.0045×10^{-3}	0.0128	0.0038	0.0140	9.183×10^{-15}	0.0120
FET06	0.0056	0.0126	0.1397×10^{-3}	0.0121	0.0012	0.0117	7.995×10^{-9}	0.0111
FET07	0.0012	0.0089	0.0394×10^{-3}	0.0086	0.0004	0.0085	2.580×10^{-10}	0.0082
FET08	0.0038	0.0098	0.0935×10^{-3}	0.0095	0.0019	0.0104	1.268×10^{-13}	0.0093
FET09	0.0062	0.0118	0.0671×10^{-3}	0.0113	0.0027	0.0110	2.213×10^{-15}	0.0092
FET10	0.0015	0.0104	0.0404×10^{-3}	0.0100	0.0023	0.0107	2.356×10^{-15}	0.0101
FET11	0.0134	0.0133	0.0068×10^{-3}	0.0128	0.0007	0.0146	8.215×10^{-15}	0.0116
FET12	0.0004	0.0109	0.0183×10^{-3}	0.0107	0.0028	0.0139	5.100×10^{-13}	0.0119
FET13	0.0029	0.0108	0.0545×10^{-3}	0.0104	0.0010	0.0122	6.045×10^{-15}	0.0108
FET14	0.0060	0.0113	0.0293×10^{-3}	0.0108	0.0007	0.0101	9.123×10^{-9}	0.0101
FET15	0.0079	0.0102	0.1083×10^{-3}	0.0098	0.0063	0.0009	6.182×10^{-8}	0.0110
FET16	0.0083	0.0118	0.1523×10^{-3}	0.0112	0.0088	0.0108	5.653×10^{-16}	0.0099

Legend: FET: Field Effect Transistor, μ_{fss}: mean of the FSS, σ_{fss}: SD of the FSS, $\mu_{me\text{-}fss}$: mean of the ME calculated from FSS, $\sigma_{me\text{-}fss}$: SD of the ME calculated from FSS, μ_{sss}: mean of the SSS, σ_{sss}: SD of the SSS, $\mu_{me\text{-}sss}$: mean of the ME calculated from SSS, $\sigma_{me\text{-}sss}$: SD of the ME calculated from SSS.

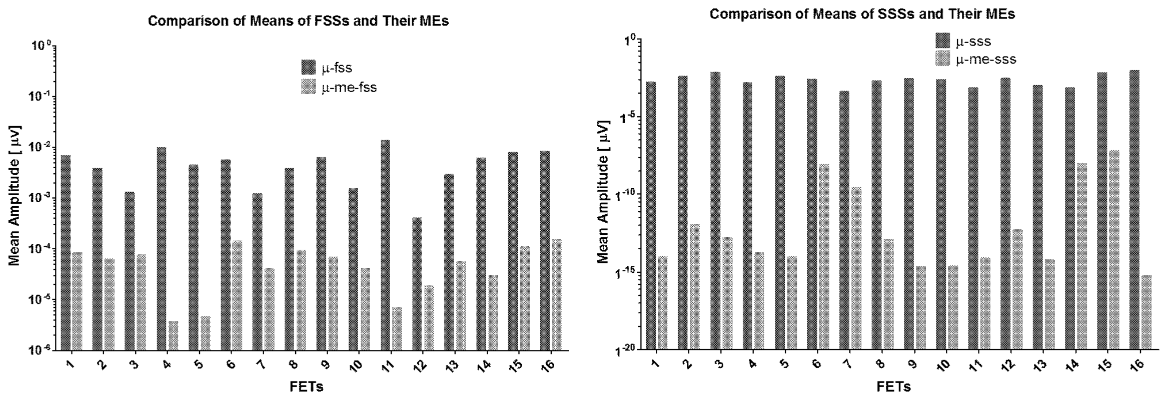


Figure 13: Graphs showing means of FSSs and their MEs (left); SSSs and their MEs (right). The y–axis scale is $\log_{10}$ based.

Figure 13 compares the means of FSS and SSS with their respective MEs from another dataset. The results are comparable to table 2, i.e., the means of the MEs were significantly small compared to those of the steady–states (the y–axis scale is $\log_{10}$ based). Also, their standard deviations were stable in the range of 0.01 to 0.015.

Furthermore, the means of the averaged steady–states (obtained by averaging the FSSs and the SSSs across a number of signals) and averaged MEs (obtained by averaging the MEs of FSSs and the MEs of SSSs across a number of signals) were very close to zero and their standard deviations were steady around 0.012 (similar to the ones calculated from single signals) as seen in figure 14.

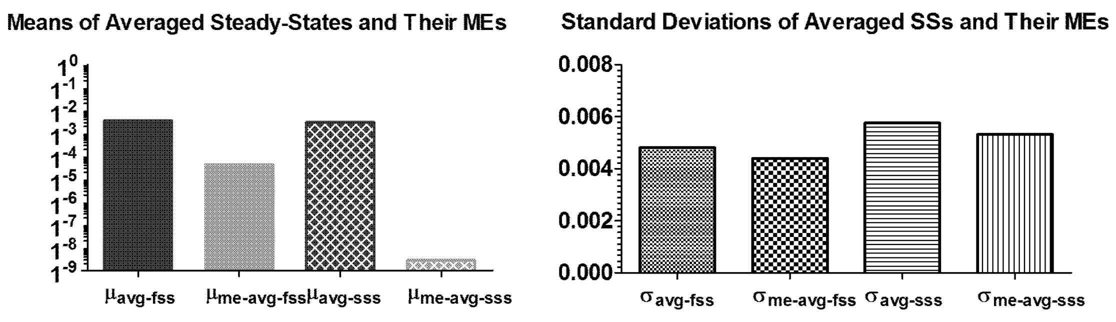


Figure 14: Left: means of averaged FSS and SSS with their respective MEs. Right: Standard deviations of Averaged FSS, SSS, and their respective MEs.

The results presented above illustrate that the MEs' means are almost zero and the signals satisfy the assumption of Gaussianity, thus implying a good quality of the recorded signals. Therefore, the noise characterization and signal quality assessment performed by the module helps the user to estimate signal quality easily and quickly. Also, analyses presented above show the module's reliable workability for cortical signals.

4.3. Latency calculation and layer activation order detection

The module was applied to a number of datasets recorded using borosilicate micropipettes and implantable EOSFET based chips. We report here results obtained by applying the method on signals recorded using micropipettes. The method found to be working well except a few situations (2% of occurrence rate) with an error of ± 300 μs in latency calculation. Particularly, this error occurred in case of signals containing slow stimulus artifacts (with frequency components less than 250 Hz). As calculated latencies were in terms of a few milliseconds up to hundreds of milliseconds, this error can be considered negligible. Figure 15 shows representative signals and their respective detected events after a run of the method.

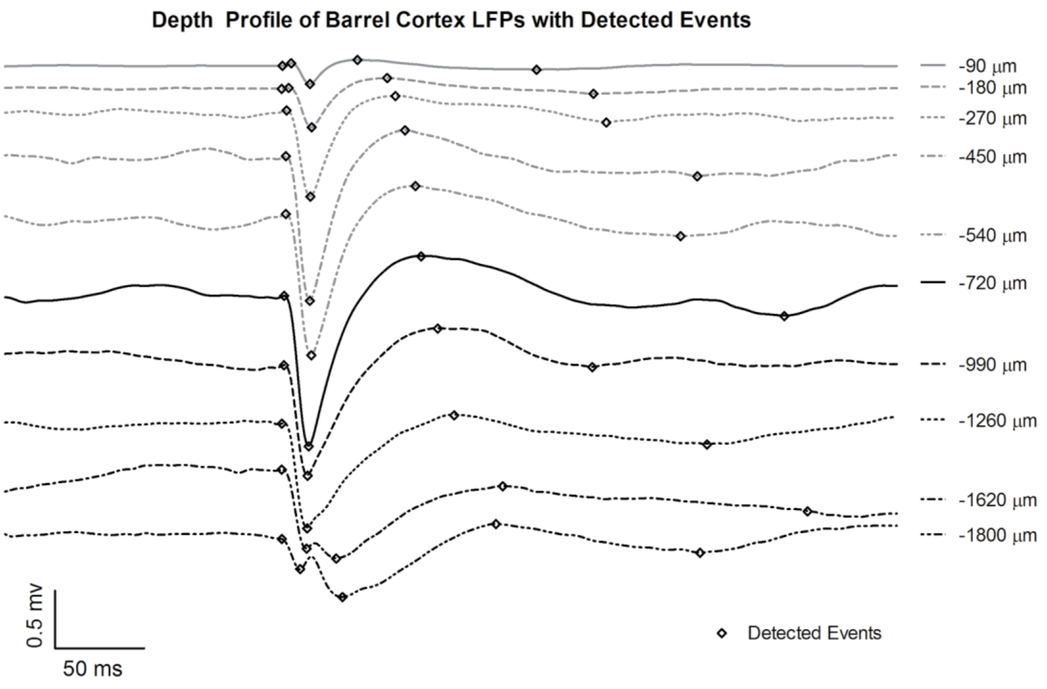


Figure 15: LFP depth profile with detected events using the method. The signals were recorded equidistantly (90 µm pitch). For better visualization only representative signals from each layer are shown.

To check the accuracy of the automated latency calculation, the LFP based latencies were also compared with the manually calculated latencies and the results were found to be similar (in table 3). ‘M’ denotes manual computation by hand and ‘A’ denotes automated calculation using the method. In table 3 the ‘E1’, ‘E2’, ‘E3’ and ‘E4’ are the latencies of the respective events. Furthermore, table 4 reports average latencies for 3 different experiments evaluated manually and by the program with their root mean square errors (RMSE). In table 4 the ‘E1’, ‘E2’, ‘E3’ and ‘E4’ are averaged latencies and RMSE of the respective events. The low RMSE indicates that the calculation of latencies using the automated method is accurate. The tables report data corresponding to representative signal(s) from depth(s) within each layer.

Table 3: Comparison of manual and automatic calculation of latencies

Depth	Mode	Latencies (ms)			
		E1	E2	E3	E4
90 µm	M	5.384	19.784	42.934	144.954

	A	5.655	19.564	42.742	143.393
180 µm	M	Absent	19.745	60.055	174.215
	A	Absent	19.416	59.259	174.023
270 µm	M	Absent	19.905	64.795	180.965
	A	Absent	19.615	63.513	183.733
450 µm	M	Absent	20.215	69.395	232.835
	A	Absent	20.228	70.32	232.836
540 µm	M	Absent	20.075	74.205	221.595
	A	Absent	20.216	74.124	222.021
720 µm	M	Absent	20.645	79.895	283.305
	A	Absent	20.565	78.228	282.532
990 µm	M	Absent	19.375	87.805	220.125
	A	Absent	19.464	87.887	175.475
1260 µm	M	Absent	18.585	96.025	238.595
	A	Absent	18.213	96.046	239.489
1620 µm	M	16.115	38.925	110.835	202.635
	A	16.116	38.785	112.562	198.448
1800 µm	M	10.175	38.585	118.825	234.975
	A	10.311	38.584	118.568	234.784

Table 4: Average latencies of events using manual and automatic calculation with RMSE

Depth	**Mode**	**Average Latencies (ms)**				**RMS Errors (ms)**			
		E1	**E2**	**E3**	**E4**	**E1**	**E2**	**E3**	**E4**
90 µm	M	6.019	19.784	42.450	139.014	0.542	0.081	0.024	0.315
	A	6.592	19.564	42.201	140.047				
180 µm	M	Absent	19.745	67.547	178.850	Absent	0.092	0.021	0.221
	A	Absent	19.416	68.974	175.654				
270 µm	M	Absent	28.517	62.574	183.015	Absent	0.026	0.032	0.254
	A	Absent	28.428	65.051	187.373				
450 µm	M	Absent	25.591	74.102	221.301	Absent	0.062	0.028	0.253
	A	Absent	25.675	77.108	203.952				
540 µm	M	Absent	18.175	71.214	231.595	Absent	0.059	0.046	0.477
	A	Absent	18.318	72.980	213.741				
720 µm	M	Absent	20.145	72.985	210.745	Absent	0.048	0.094	0.351
	A	Absent	19.619	73.428	271.659				
990 µm	M	Absent	21.937	84.862	192.251	Absent	0.095	0.392	0.853
	A	Absent	22.121	90.957	183.241				
1260 µm	M	Absent	18.985	91.213	210.021	Absent	0.036	0.095	0.764
	A	Absent	19.018	91.478	228.674				
1620 µm	M	11.152	26.132	110.835	192.380	0.152	0.071	0.93	0.429
	A	10.920	25.925	112.562	181.154				
1800 µm	M	9.631	35.585	117.241	221.341	0.821	0.087	0.034	0.762

A	9.927	35.885	113.231	214.114

From the CSD profile the latencies were calculated as the difference between the first sink's peak and stimulus–onset. Figure 16 shows a comparison of the cortical layer activation order using LFPs and CSDs (Mahmud et al., 2011a).

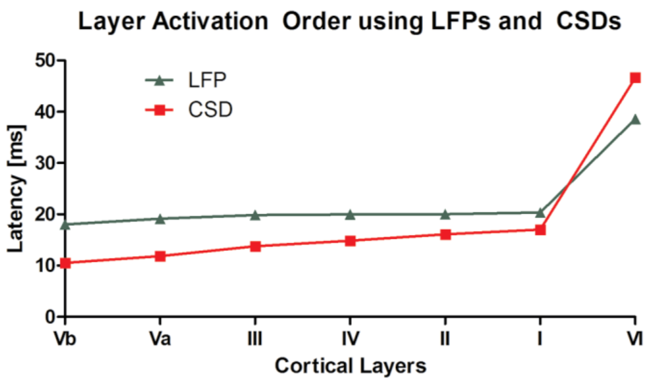


Figure 16: Comparison of cortical layer activation order obtained from LFPs and CSDs.

Basing on these evidences, we can assert that the latency estimation module can calculate the latencies, and, in turn, that the activation order of layers in the barrel columns is calculated accurately. Both the approaches (using LFP or CSD) provide similar results which are justifiable using a simple barrel cortex network model from Fox, 2008. Thus, it is the user's choice which one of the two approaches to use (Mahmud et al., 2011a).

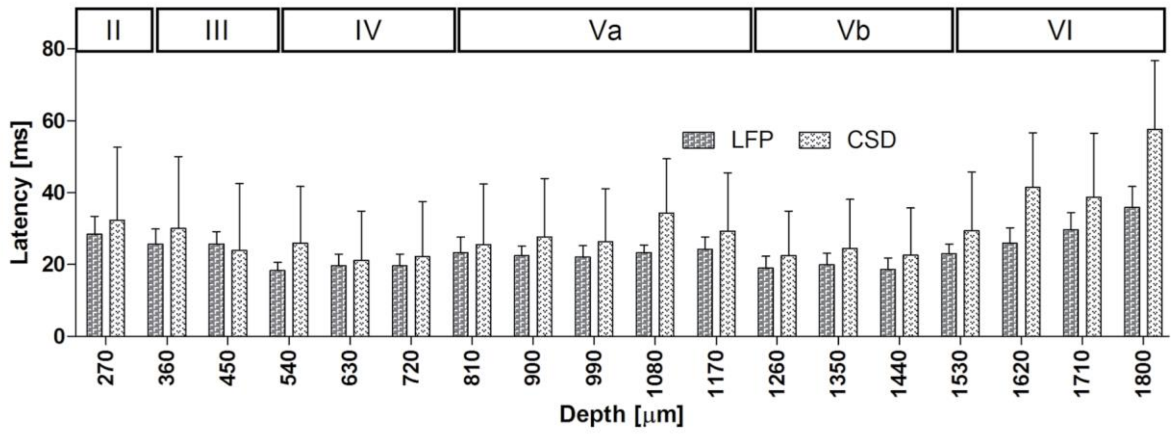


Figure 17: Layer activation order using latencies calculated from LFPs averaged across experiments and CSDs calculated from them are depicted in (a) and (b), respectively.

Application of the method on averaged signals across experiments (*n*=3) provided a temporal order of layer activation comparable with previous studies done by Armstrong-James et al., 1992; Di et al., 1990; and Einevoll et al.,2007. Figure 17 shows the layer activation order using LFPs averaged across experiments and CSDs derived from them, respectively.

4.4. Clustering of single LFPs

The module was validated on a number of datasets recorded with borosilicate micropipettes. Results were found satisfactory except some cases, where signal morphology was highly atypical (occurrence rate of 1%). Each dataset comprised recordings from about 20 different depths, and each of them contained as many as 100 single sweeps. Except to demonstrate the distribution of single LFPs to different clusters, we present clustering results related to a representative set of single LFPs.

In figure 18 we can see the raw single LFPs and their average signal (left), and the estimated single sweeps and their average signal (right).

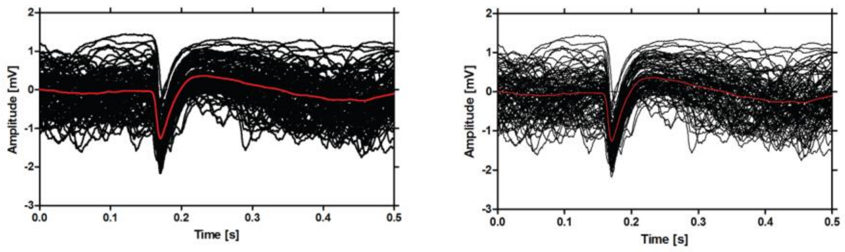


Figure 18: Left: raw sweeps (without estimation) with average in red. Right: estimated sweeps with average in red. The noise in the raw single sweeps is evident in the left figure.

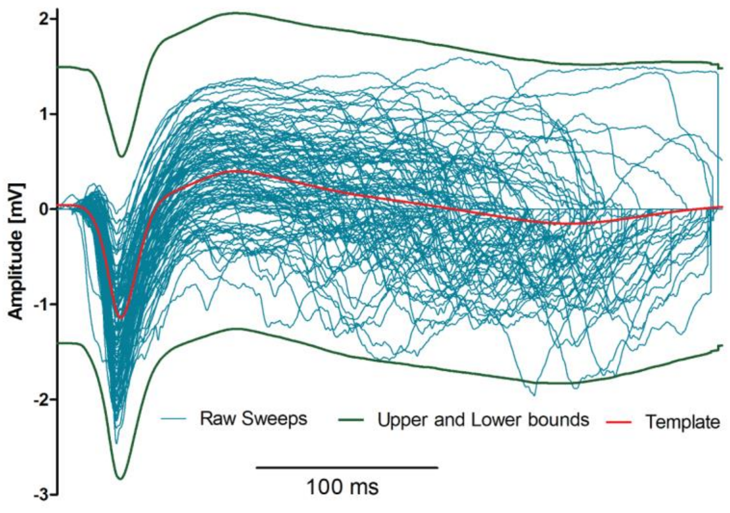


Figure 19: The template (in red), the upper and lower bounds (in green), and the single sweeps truncated to the size of the template. It also shows the recognized single LFPs using contour comparison of the template (blue).

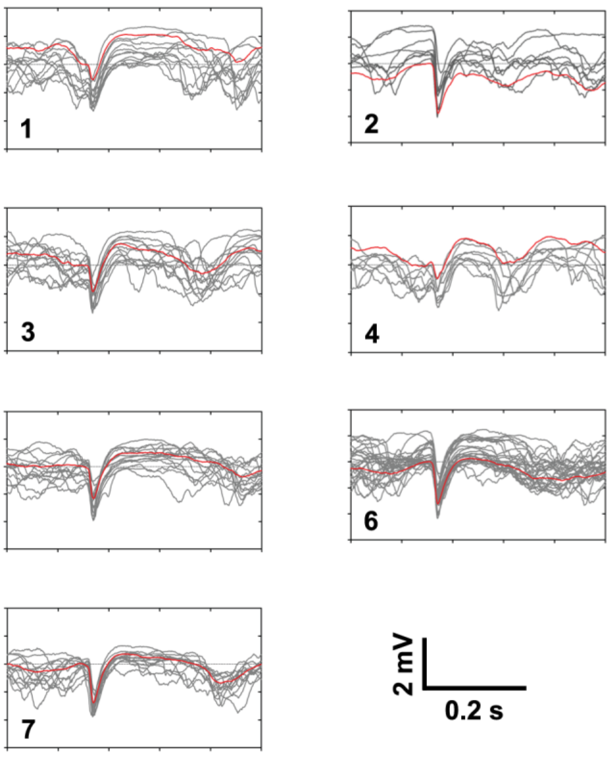


Figure 20: Result of the clustering. Single sweeps (in grey) and their respective averages (in red) depict the clear difference in the inter–cluster signal shapes.

Figure 19 shows single sweeps truncated to the size of template, the upper and lower bounds of the template. Each single LFP that fell within these bounds was considered to be recognized.

Figure 20 shows the various clusters of signals with their respective averages. The recognized N single LFPs were classified using the *i*K–means clustering method to obtain different shape based clustering.

Table 5: Total Recognized Sweeps and Single LFP Allocation to Clusters

Depth	**RS**	**Clusters**									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
90 µm	90	5	6	12	11	11	7	8	12	6	12
180 µm	86	11	17	10	17	18	13	–	–	–	–
270 µm	87	8	8	8	11	15	7	4	10	10	6
360 µm	80	7	10	7	14	13	9	11	9	–	–
450 µm	78	10	9	11	4	7	15	9	13	–	–
540 µm	86	9	8	16	9	9	2	9	8	7	9
630 µm	85	16	6	15	14	17	17	–	–	–	–
720 µm	93	6	18	17	16	7	16	13	–	–	–
810 µm	92	10	9	13	9	13	8	14	6	10	–
900 µm	97	11	15	6	8	14	10	6	9	9	9
990 µm	96	19	15	15	10	5	17	15	–	–	–
1080 µm	92	12	12	9	9	12	9	8	10	11	–
1170 µm	99	8	13	13	9	6	11	10	9	10	10
1260 µm	100	11	20	13	16	7	7	16	10	–	–
1350 µm	100	18	13	19	19	19	12	–	–	–	–
1440 µm	98	13	10	16	8	16	14	7	5	9	–
1530 µm	100	7	9	18	7	14	7	10	16	12	–
1620 µm	99	10	5	16	10	11	8	10	12	11	6
1710 µm	99	10	12	20	13	14	12	18	–	–	–
1800 µm	100	10	12	5	15	16	14	9	19	–	–

Table 5 tabulates the recording depths, total number of recognized LFPs, and single LFP signal distribution among different clusters. This table shows that the single LFPs were well classified into different clusters. In the table, 'RS' denotes total number of recognized signals in one recording position, '1' to '10' are the cluster numbers, and '–' means no clusters

Once the single sweep clusters were formed, the program computes local averages of each cluster for further processing. Analyses of these local averages (the signal amplitudes of the E2 and the calculated latencies based on the signal events) revealed that the activation of underlying neuronal network generating the signal might be different even if the signals were recorded from the same recording site with the same stimulus (figure 21 and figure 22).

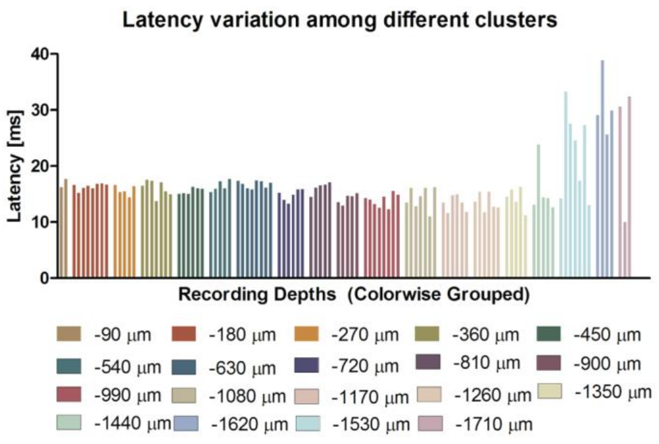


Figure 21: Latency variation among different clusters local averages. Each bar corresponds to a local average of a cluster and each color corresponds to a recording depth consisting of a number of clusters.

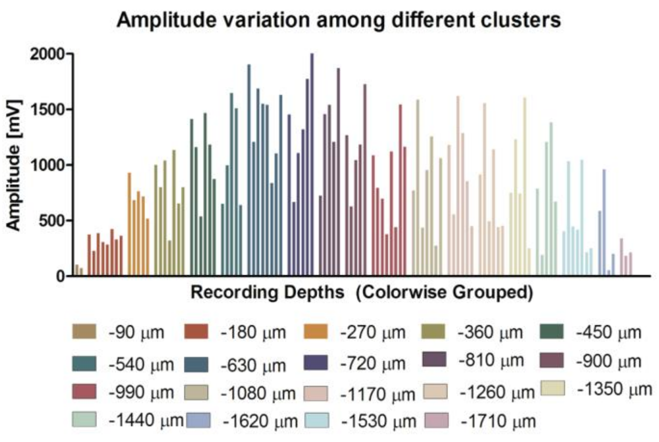


Figure 22: Amplitude variation among different clusters local averages.

From these variations in the latency and the amplitude it can be asserted that different neuronal networks near the recording electrode were activated during the whisker stimulation at different times.

5. Conclusion

To understand brain activities with an unprecedented level, parallel high resolution recordings are required. This paper presents a report on the SigMate software package. As with the growth of multisite neuronal probes, amount of acquired data are increasing, the need of one single software package performing all necessary processing and analysis on the data has become crucial. This is the first step towards meeting that need. As the software has been extensively tested with two possible sources of data, we believe that once it is disseminated to the community (which will happen in the near future) it will serve a good deal in analyzing extracellular neurophysiological signals (Mahmud et al., 2010b; 2011b). Modules for coherence and correlation based analysis to obtain more information about the brain’s functionality from the recorded signals, neuronal network based on different stimuli to be able to predict the signals a priori and compare them with the recorded signals, and understand the activation of underlying neuronal networks generating the signals are under development. Also, in the near future we will convert SigMate to perform online signal processing and analysis.

Acknowledgments

This work was carried out as a part of the European Commission funded CyberRat project under the Seventh Framework Programme (ICT-2007.8.3 Bio-ICT convergence, 216528, CyberRat). We express our sincere gratitude to the anonymous referees for their fruitful comments to improve the quality of the paper.

A. Algorithms

A.1. Peak–valley detection

The following algorithm is used for the detection of peaks–and–valleys of the signal to have an estimation of the artifact signal.

Function: Detect Peak-Valley()

Input: Signal file, whose peaks and valleys are to be found.

Output: Peaks-and-valleys of the input signal.

Method:

```
1. Initialize, S:=signal; and Threshold:=Standard-deviation(S);
2. Set, Peak:=infinity; Valley=-infinity; and Flag:=True;
3. Current:=Current element of S;
4. if (Current > Peak), Reset, Peak:= Current; end if ;
5. if (Current < Peak), Reset, Valley:= Current; end if ;
6. if (Flag is True)
       if (Current <(Peak − Threshold))
           Add Current to Peaks; and
           Reset, Flag:=False;
       end if ;
       if (Current >(Valley + Threshold))
           Add Current to Valleys; and
           Reset, Flag:=True;
       end if;
   end if;
7. Repeat step 3 to 6 for every element of the signal.
8. Return Peaks and Valleys.
```

A.2. Calculation of measurement errors

The following algorithm is used in calculating the MEs. As described in section 3.4, the MEs are calculated after having detected the FSS and the SSS. Thus, this method calls two other methods which detect the FSS and the SSS.

Function: MeasurementErrors()

Input: Signal files containing time and data.

Output: Statistical information (μ, σ and distribution) of the data.

Method:

// this is for the single sweep approach.

```
1. for each signal file
    a. Load time(t) and data(s);
    b. firstSteadyState[fssT, fssS]:=Call the findFirstSteadyState(t, s);
```

```
        c. secondSteadyState[sssT, sssS]:=Call the findSecondSteadyState(t, s, firstSteadyState);
        d. Fit mathematical models to the firstSteadyState and secondSteadyState;
        e. Calculate the data bounds around the firstSteadyState and secondSteadyState;
        f. Box data points based on the data bound, i.e., boxing the data around $\pm 2\sigma$ from the straight line;
        g. Calculate MEs for first and SSSs using equation(1);
        h. Characterize MEs by calculating $\mu$ and $\sigma$;
        i. Plot the distribution of MEs using histogram and estimate the distribution using smoothing function;
    end for
// this is for the averaged steady–state approach.
    2. Calculate the average of the all FSSs;
    3. Fit mathematical model to the average of the FSSs;
    4. Calculate the ME of the average using equation (1);
    5. Characterize error by calculating $\mu$ and $\sigma$;
    6. Calculate the average of the all SSSs;
    7. Fit mathematical model to the average of the SSSs;
    8. Calculate the ME of the average using equation (1);
    9. Characterize error by calculating $\mu$ and $\sigma$;
    10. Plot the distribution of ME using histogram and estimate the distribution using a smoothing function;
// this is for the averaged ME approach.
    11. Calculate average of all MEs for the FSS;
    12. Characterize error by calculating $\mu$ and $\sigma$;
    13. Plot the distribution using histogram and estimate the distribution using a smoothing function;
    14. Calculate average of all MEs for the SSS;
    15. Characterize error by calculating $\mu$ and $\sigma$;
    16. Plot the distribution using histogram and estimate the distribution using a smoothing function;
```

A.2.1. Detecting the first–steady–state

The following algorithm is used in detecting the first–steady state.

Function: findFirstSteadyState()

Input: The time(t) and data(s).

Output: The FSS of the signal (firstSteadyState).

Method:

```
1. firstPart:=the first 10 ms of the signal, s;
2. stdFirst:=std(firstPart); stdS:=std(s);
3. intervalInt:= 3 ms; newS:=rest of the signal;
4. Divide newS into intervals of length intervalInt;
5. Initialize, newPartS:=[]; newPartT:=[]; flag:=1;
6. while currInterval isn't in evoked response
```

```
    fitLine:=Fit a straight line to currInterval;
    sdNew:=std(currInterval);
    if (sdNew ≥ stdCheck && flag)
        add currInterval to newPartS;
        add currIntervalTime to newPartT;
        stdCheck:= sdNew; flag:= 1;
    else
        flag:=0;
        add currInterval to newPartS;
        add currIntervalTime to newPartT;
        if (last point of fitLine > stdS)
            remove currInterval from newPartS;
            remove currIntervalTime from newPartT;
        end if;
    end if;
  end while;
7. Return firstSteadyState:=[newPartT,newPartS];
```

A.2.2. Detecting second–steady–state

The following algorithm detects and returns the second–steady–state.

Function: findSecondSteadyState(t, s,fsState)

Input: Signaltime(t) anddata(s), and theFSS(fsState).

Output: TheSSS ofthe signal(secondSteadyState).

Method:

```
1. fsStateLength:=length(fsState); fsStateStd:=standard deviation(fsState);
2. sToAnalyse:=s(fsStateLength to length(s));
3. sToAnalyseT:=t(fsStateLength to length(s));
4. numDiv:=floor(length(s)/fsStateLength);
5. sToAnalyseRev:=Reverse(sToAnalyse);
6. sToAnalyseRevT:=Reverse(sToAnalyseT);
7. Initialize, ssStateS:=[]; ssStateT:=[]; flag:=0;
8. Find which part of sToAnalyseS has to be considered as SSS
    for i:=1 to numDiv
        currInterval:=sToAnalyseRev[i];
        currIntervalTime:=sToAnalyseRevT[i];
        stdCurrInterval:=std(currInterval);
        if stdCurrInterval < fsStateStd
```

```
            add the currInterval to ssStateS;
            add the currIntervalTime to ssStateT;
        else
            if (i==1)
                add currInterval to ssStateS;
                add the currIntervalTime to ssStateT;
            end if;
            flag:=flag+1;
            if (flag<=2) &&(length(sToAnalyseRev) < length(fsState))
                nextInterval:= sToAnalyseRev[i+1];
                nextIntervalT:=sToAnalyseRevT[i+1];
                stdNextInterval:=std(nextInterval);
                if (stdNextInterval <= fsStateStd)
                    add the nextInterval to ssStateS;
                    add the nextIntervalT to ssStateT;
                end if;
            else
                exit the loop;
            end if;
        end if;
    end for;
9. sssRevT:=Reverse(ssStateT); sssRevS:=Reverse(ssStateS);
10. Return secondSteadyState:=[sssRevT, sssRevS];
```

References

Ahissar E, Knutsen KM. Object localization with whiskers. Biol Cybern 2008; 98(6): 449–58.

Ahrens KF, Kleinfeld D. Current flow in vibrissa motor cortex can phase– lock with exploratory rhythmic whisking in rat. J Neurophysiol, 2004; 92: 1700–7.

Akaike H. A new look at the statistical model identification. IEEE T Automat Contr, 1974; 19: 716–22.

Alloway KD. Information processing streams in rodent barrel cortex: the differential functions of barrel and septal circuits. CerebCortex,2008; 18(5): 979–98.

Armstrong-James M, Fox K, Das-Gupta A. Flow of excitation within rat barrel cortex on stiking a single vibrissa. J Neurophysiol 1992; 68(4): 1345–57.

Bock HH. Clustering methods: a history of K-Means algorithms. In: Brito P, Cucumel G, deCarvalho F, editor. Selected contributions in data analysis and classification, 2007; 888: 161–72.

Bokil HS, Andrews P, Kulkarni JE, Mehta S, Mitra PP. Chronux: A platform for analyzing neural signals. J Neurosci Methods 2010; 192: 146–51.

Bologna LL, Pasquale V, Garofalo M, Gandolfo M,Baljon PL, Maccione A, Martinoia S, Chiappalone M. Investigating neuronal activity by SPYCODE multi–channel data analyzer.

Neural Networks, 2010; 23(6): 685–97.

Bonomini MP, Ferrandez JM, Bolea JA, Fernandez E. DATA-MEAns: An open source tool for the classification and management of neural ensemble recordings. J Neurosci Methods, 2005; 148: 137–46.

Bowman AW, Azzalini A. Applied Smoothing Techniques for Data Analysis. New York: Oxford University Press; 1997.

Buzsaki G. Large-scale recording of neuronal ensembles. Nat Neurosci, 2004; 7(5): 446-51.

Castro-Alamancos MA, Oldford E. Cortical sensory suppression during arousal is due to the activity–dependent depression of thalamocortical synapses. J Physiol, 2002; 541: 319-31.

Chiang MMT, Mirkin B. Intelligent choice of the number of clusters in K– Means clustering: an experimental study with different cluster spreads. J Classif, 2010; 27(1): 3-40.

Cui J, Xu L, Bressler SL, Ding M, Liang H. BSMART: A Matlab/C toolbox for analysis of multichannel neural time series. Neural Networks, 2008; 21(8): 1094-104.

Delorme A, Makeig S. EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. J Neurosci Meth, 2004; 134: 9–21.

Di S, Baumgartner C, Barth D S. Laminar Analysis of extracellular field potentials in rat vibrissa/barrel cortex. J Neurophysiol, 1990; 63: 832-40.

Diamond ME, vo Heimendahl M, Knutsehn PM, Kleinfeld D, Ahissar D. ‘Where’ and ‘what’ in the whisker sensorimotor system. Nat Rev Neurosci, 2008; 9: 601-12.

Dodge Y. The Oxford Dictionary of Statistical Terms. New York: Oxford University Press; 2003.

Eckerson WW. Three tier client/server architecture: achieving scalability, performance, and efficiency in client server applications. Open Information Systems, 1995; 10(1): 3(20).

Egert U, Knott Th, Schwarz C, Nawrot M, Brandt A, Rotter S, Diesmann M. MEA-Tools: an open source toolbox for the analysis of multi–electrode data with Matlab. J Neurosci Methods, 2002; 117(1): 33-42.

Einevoll GT, Pettersen KH, Devor A, Ulbert I, Halgren E, Dale AM. Laminar population analysis: Estimating firing rates and evoked synaptic activity from multielectrode recordings in rat barrel cortex. J Neurophysiol2007; 97: 2174-90.

Felderer F, Fromherz P, Transistor needle chip for recording in brain tissue, Appl Phys A, 2011; 104:1-6.

Fletcher M, Liang B, Smith L, Knowles A, Jackson T, Jessop M, Austin J. Neural network based pattern matching and spike detection tools and services - in the CARMEN neuroinformatics project. Neural Networks 2008; 21: 1076-84.

Fox K. Barrel Cortex. Cambridge: Cambridge University Press; 2008.

Goldberg D, Victor J, Gardner E, Gardner D. Spike train analysis toolkit: enabling wider application of information–theoretic techniques to neurophysiology. Neuroinformatics, 2009; 7(3): 165-78.

Gunay C, Edgerton J, Li S, Sangrey T, Prinz A, Jaeger D. Database analysis of simulated and recorded electrophysiological datasets with PANDORAs toolbox. Neuroinformatics, 2009; 7(2):

93-111.

Hazan L, Zugaro M, Buzsaki G. Klusters, NeuroScope, NDManager: A free software suite for neurophysiological data processing and visualization. J Neurosci Methods, 2006; 155: 207-16.

Herz AVM, Meier R, Nawrot MP, Schiegel W, Zito T. G–Node: An integrated tool-sharing platform to support cellular and systems neurophysiology in the age of global neuroinformatics. Neural Networks, 2008; 21(8):1070-5.

Hierlemann A, Frey U, Hafizovic S, Heer F. Growing Cells Atop Microelectronic Chips: Interfacing Electrogenic Cells In Vitro With CMOS-Based Microelectrode Arrays. Proc IEEE, 2011; 99(2): 252-284.

Huang Y, Li X, Li Y, Xu Q, Lu Q, Liu Q. An integrative analysis platform for multiple neural spike train data. J Neurosci Methods, 2008; 172(2): 303-11.

Hutzler M, Lambacher A, Eversmann B, Jenkner M, Thewes R, Fromherz P. High-Resolution Multitransistor Array Recording of Electrical Field Potentials in Cultured Brain Slices. J Neurophysiol, 2006; 96: 1638-1645.

Jellema T, Brunia CHM, Wadman WJ. Sequential activation of microcircuits underlying somatosensory-evoked potentials in rat neocortex. Neuroscience, 2004; 129: 283-95.

Kaur S, Rose H J, Lazar R, Liang K, Metherate R. Spectral integration in primary auditory cortex: laminar processing of afferent input, in vivo and in vitro. Neuroscience, 2005; 134: 1033-45.

Kublik E. Contextual impact on sensory processing at the barrel cortex of awake rat. Acta Neurobiol Exp, 2004; 64: 229-38.

Kwon KY, Eldawlatly S, Oweiss KG. NeuroQuest: A comprehensive analysis tool for extracellular neural ensemble recording. J Neurosci Methods, 2011, 189-201.

Lambacher A, Vitzthum V, Zeitler R, Eickenscheidt M, Eversmann B, Thewes R, Fromherz P. Identifying firing mammalian neurons in networks with high-resolution multi-transistor array (MTA). Appl Phys A, 2011; 102: 1-11.

Landlaw EM, DiStefano Third JJ. Multiexponential, multicompartmental, and noncompartmental modeling. II. Data analysis and statistical consideration. Am J Physiol Regul Integr Comp Physiol, 1984; 246: R665-77.

Legatt A, Arezzo J, Vaughan HG. Averaged multiple unit activity as an estimate of phasic changes in local neuronal activity: effects of volume-conducted potentials. J Neurosci Meth, 1980; 2(2): 203-17.

Lidierth M. sigTOOL: A Matlab–based environment for sharing laboratory developed software to analyze biological signals. J Neurosci Methods, 2009; 178: 188-96.

Macqueen J. Some methods for classification and analysis of multivariate observations. Proc of the Fifth Berkeley Symp on Math Statist and Prob, 1967; 1: 281-97.

Madsen K, Nielsen HB, Tingleff O. Methods for Non-Linear Least Squares Problems. Denmark: Informatics and Mathematical Modelling, Technical University of Denmark (DTU), Kgs. Lyngby; 2004.

Magri C, Whittingstall K, Singh V, Logothetis N, Panzeri S. A toolbox for the fast information analysis of multiple-site LFP, EEG and spike train recordings. BMC Neuroscience, 2009; 10(1): 81.

Mahmud M, Bertoldo A, Maschietto M, Girardi S, Vassanelli S. Automatic detection of layer activation order in information processing pathways of rat barrel cortex under mechanical whisker stimulation. In: Proc of the IEEE EMBC2010; 2010a. p. 6095-8.

Mahmud M, Bertoldo A, Girardi S, Maschietto M, Vassanelli S. SigMate: a Matlab–based neuronal signal processing tool. In: Proc. of the IEEE EMBC2010; 2010b. p. 1352-5.

Mahmud M, Travalin D, Bertoldo A, Girardi S, Maschietto M, Vassanelli S. A contour based automatic method to classify local field potentials recorded from rat barrel cortex. In: Proc. 5th Cairo Intl. Conf. Biomedical Eng. (CIBEC2010); 2010c. p. 163-6

Mahmud M, Girardi S, Maschietto M, Rahman MM, Bertoldo A, Vassanelli S. Slow stimulus artifact removal through peak-valley detection of neuronal signals recorded from somatosensory cortex by high resolution brain-chip interface. In: IFMBE Proceedings - WC2009, Germany, 2009a; p. 2062-5.

Mahmud M, Girardi S, Maschietto M, Rahman MM, Bertoldo A, Vassanelli S. Noise characterization of electrophysiological signals recorded from high resolution brain–chip interface. In: Proc of the ISBB2009, Melbourne, Australia, 2009b; pp. 84-7.

Mahmud M, Pasqualotto E, Bertoldo A, Girardi S, Maschietto M, Vassanelli S. An automated method for detection of layer activation order in information processing pathway of rat barrel cortex under mechanical whisker stimulation. J Neurosci Meth, 2011a; 196: 141-150.

Mahmud M, Bertoldo A, Girardi S, Maschietto M, Pasqualotto E, Vassanelli S. SigMate: a comprehensive software package for extracellular neuronal signal processing and analysis. In: Proc. of 5th Intl. IEEE EMBS Conf. Neural Eng. (NER2011); 2011b. p. 88-91.

Mahmud M, Travalin D, Bertoldo A, Girardi S, Maschietto M, Vassanelli S. An automated classification method for single sweep local field potentials recorded from rat barrel cortex under mechanical whisker stimulation. J. Med. Biol. Eng., 2011c; doi: 10.5405/jmbe.923 (in press).

Mahmud M, Bertoldo A, Girardi S, Maschietto M, Vassanelli S. An automated quality assessment method for cortical signals recorded by high resolution brain-chip interface from S1 brain cortex. unpublished, 2011d.

Megevand P, Troncoso E, Quairiaux C, Muller D, Michel CM, Kiss JZ. Long-term plasticity in mouse sensorimotor circuits after rhythmic whisker stimulation. J Neurosci, 2009; 29(16): 5326-35.

Meier R, Egert U, Aertsen A, Nawrot MP. FIND–A unified framework for neural data analysis. Neural Networks, 2008; 21(8): 1085-93.

Mitzdorf U, Singer W. Monocular activation of visual cortex in normal and monocularly deprived cats: an analysis of evoked potentials. JPhysiol, 1980; 304: 203-20.

Mitzdorf U. Current source-density method and application in cat cerebral cortex: investigation of evoked potentials and EEG phenomena. Physiol Rev, 1985; 65: 37-100.

Morup M, Hansen LK, Arnfred SM. ERPWAVELAB: A toolbox for multi-channel analysis of time-frequency transformed event related potentials. J Neurosci Meth, 2007; 161(2): 361-8.

NDF. Available at: http://www.carmen.org.uk/standards/CarmenDataSpecs.pdf (Retrieved on March 15, 2012)

Neuroshare. http://www.neuroshare.org/ (Retrieved on February 6, 2012.)

Novellino A, Chiappalone M, Maccione A, Martinoia S. Neural Signal Manager: a collection of classical and innovative tools for multi-channel spike train analysis. Int J Adapt Control Signal Process, 2009; 23(11): 999-1013.

Okun M, Naim A, Lampl I. The subthreshold relation between cortical local field potential and neuronal firing unveiled by intracellular recordings in awake rats. J Neurosci, 2010; 30(12): 4440-8.

Pettersen KH, Devor A, Ulbert I, Dale AM, Einevoll GT. Current–source density estimation based on inversion of electrostatic forward solution: Effects of finite extent of neuronal activity and conductivity discontinuities. J Neurosci Meth, 2006; 154: 116-33.

Prochazka A, Mushahwar VK, McCreery DB. Neuralprostheses. J Physiol, 2001; 533(Pt1): 99-109.

Quiroga RQ, Nadasdy Z, Ben-Shaul Y. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. Neural Computation 2004; 16(8): 1661-87.

Rappelsberger P, Pockberger H, Petsche H. Current source density analysis: methods and application to simultaneously recorded field potentials of the rabbit's visual cortex. Pflugers Arch, 1981; 389: 159-70.

Smith LS, Mtetwa N. A tool for synthesizing spike trains with realistic interference. J Neurosci Meth 2007; 159: 170-80.

Staba RJ, Bergmann PC, Barth DS. Dissociation of slow waves and fast oscillations above 200 Hz during GABA application in rat somatosensory cortex. J Physiol, 2004; 561(1): 205-14.

Swadlow H A, Gusev A G, Bezdudnaya T. Activation of a cortical column by a thalamocortical impulse. J Neurosci, 2002; 22(17): 7766-73.

Szymanski FD, Garcia-Lazaro JA, Schnupp JWH. Current source density profiles of stimulus-specific adaptation in rat auditory cortex. J Neurophysiol, 2009; 102: 1483-90.

van Hemmen J, Ritz R. Neural coding: A theoretical vista of mechanisms, techniques, and applications. In: Andersson SI, editor. Lecture Notes in Computer Science, Analysis of Dynamical and Cognitive Systems, 1995; 888: 75-119.

Vargas-Irwin C, Donoghue JP. Automated spike sorting using density grid contour clustering and subtractive waveform decomposition. J Neurosci Meth, 2007; 164: 1-18.

Vassanelli S, Fromherz P. Neurons fromrat brain coupled to transistors. Appl Phys A, 1997; 65: 85-88.

Vassanelli S, Fromherz P. Transistor records of excitable neurons from rat brain. Appl Phys A, 1998; 66: 459-463.

Vassanelli S, Fromherz P. Transistor Probes Local Potassium Conductances in the Adhesion Region of Cultured Rat Hippocampal Neurons. J Neurosci, 1999; 19(16): 6767-73.

Vassanelli S, Mahmud M, Girardi S, Maschietto M. On the Way to Large-Scale and High-Resolution Brain-Chip Interfacing. Cogn Comput, 2012; 4(1): 71-81.

Vato A, Bonzano L, Chiappalone M, Cicero S, Morabito F, Novellino A, Stillo G. Spike manager: a new tool for spontaneous and evoked neuronal networks activity characterization. Neurocomputing, 2004; 58-60: 1153-61.

Versace M, Ames H, Lveill J, Fortenberry B, Gorchetchnikov A. KInNeSS: a modular framework for computational neuroscience. Neuroinformatics, 2008; 6(4): 291-309.

Wagenaar D, DeMarse TB, Potter SM. MeaBench: A toolset for multi-electrode data acquisition and on-line analysis. In: Proc IEEE EMBS Conf on Neural Eng, 2005; pp. v-viii.

Watson P, Hiden H, Woodman S. e-Science Central for CARMEN: science as a service. Concurrency Computat: Pract Exper 2010; 22: 2369-80.

Wise KD, Anderson DJ, Hetke JF, Kipke DR, Najafi K. Wireless implantable microsystems: high-density electronic interfaces to the nervous system. Proc IEEE, 2004; 92: 76-97.