Neural Networks 23 (2010) 685–697

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2010 Special Issue

# Investigating neuronal activity by SPYCODE multi-channel data analyzer

Luca Leonardo Bologna[a,c], Valentina Pasquale[a], Matteo Garofalo[a], Mauro Gandolfo[b], Pieter Laurens Baljon[b,d], Alessandro Maccione[a], Sergio Martinoia[a,b], Michela Chiappalone[a,*]

[a] Department of Neuroscience and Brain Technologies, Italian Institute of Technology — IIT, Via Morego 30, 16163, Genova, Italy

[b] Neuroengineering and Bio-nano Technology Group (NBT), Department of Biophysical and Electronic Engineering (DIBE), University of Genova, Via all'Opera Pia 11A, 16145, Genova, Italy

[c] Laboratory of Neurobiology of Adaptive Processes, University Pierre & Marie Curie, 9 quai St. Bernard, 75005 Paris, France

[d] Broad Fellows Program and Division of Biology, California Institute of Technology, 1200 E. California Blvd, 91125-0076 Pasadena, CA, USA

ARTICLE INFO

*Article history:*
Received 9 October 2009
Received in revised form 29 March 2010
Accepted 5 May 2010

*Keywords:*
Micro-electrode array
Cell culture
Batch processing
Connectivity

ABSTRACT

Multi-channel acquisition from neuronal networks, either *in vivo* or *in vitro*, is becoming a standard in modern neuroscience in order to infer how cell assemblies communicate. In spite of the large diffusion of micro-electrode-array-based systems, researchers usually find it difficult to manage the huge quantity of data routinely recorded during the experimental sessions. In fact, many of the available open-source toolboxes still lack two fundamental requirements for treating multi-channel recordings: (i) a rich repertoire of algorithms for extracting information both at a single channel and at the whole network level; (ii) the capability of autonomously repeating the same set of computational operations to 'multiple' recording streams (also from different experiments) and without a manual intervention. The software package we are proposing, named SPYCODE, was mainly developed to respond to the above constraints and generally to offer the scientific community a 'smart' tool for multi-channel data processing.

## 1. Introduction

The brain is undoubtedly the most fascinating but still mysterious machine of the known universe. The Nobel laureate Santiago Ramon Y. Cajal used to say that 'the brain is a world consisting of a number of unexplored continents and great stretches of unknown territory'. After more than one hundred years, his sentence remains true but the progress in technological research makes modern neuroscientists optimistic about the possibility to open an access towards those 'unexplored' territories. Among others, recent advancements in micro-fabrication technologies enabled the introduction of devices for multi-channel recordings (i.e. Micro-Electrode Arrays, MEAs), giving the capability of investigating neural interdependency and computational properties of dynamically interacting cell assemblies (Hebb, 1949; Miller & Wilson, 2008; Quiroga & Panzeri, 2009).

Thanks to the pioneering work by Gross (Gross, Azzazy, Wu, & Rhodes, 1995; Gross, Rhoades, & Jordan, 1992; Gross, Rieske, Kreutzberg, & Meyer, 1977), who first demonstrated the possibility to use neuronal cultures coupled to MEAs as a cell-based biosensor, MEA-based devices are now a well-accepted electrophysiological technique for both *in vivo* (Blanche, Spacek, Hetke, & Swindale, 2005; Buzsaki, 2004) and *in vitro* (Egert, Heck, & Aertsen, 2002; Minerbi et al., 2009; Schneidman, Berry, Segev, & Bialek, 2006) measurements. Tens of micro-electrodes permanently in contact with electrogenic cells allow monitoring the electrophysiological activity of a cell population (i.e., multi-units recordings) for long periods of time (Bologna et al., 2010; Chiappalone, Bove, Vato, Tedesco, & Martinoia, 2006; Wagenaar, Pine, & Potter, 2006). Such a system represents a perfect candidate to routinely record and evaluate the patterns of spontaneous as well as stimulated network behavior (Bakkum, Chao, & Potter, 2008; Chiappalone, Massobrio, & Martinoia, 2008; Marom & Shahaf, 2002).

MEA-based systems are currently available on the market (Multi Channel Systems, Reutlingen, Germany; Panasonic, Osaka, Japan; Ayanda Biosystems, Lausanne, Switzerland; Plexon, Dallas TX, USA; Axion Biosystems, Atlanta, GA, USA) and find applications in many research fields, such as neuroscience, pharmacology, cardiac electrophysiology, neurorobotics and bidirectional brain machine interfaces (Nicolelis, 2003). Notwithstanding the widespread use of this technique, there is still a lack of efficient software to manage a large amount of electrophysiological data produced by such multi-site recordings (Nicolelis, 2001; Potter, 2001). Especially in the case of neuropharmacological experiments, tens of gigabytes of data are daily produced in order to characterize the response to specific drugs or to test the effects of unknown ones (Gramowski et al., 2006; Gramowski, Jugelt, Weiss, & Gross, 2004).

Among the general-purpose commercially available processing tools, often sold together with the acquisition system (e.g.

* Corresponding author. Tel.: +39 010 71781 743; fax: +39 010 720321.
*E-mail address:* michela.chiappalone@iit.it (M. Chiappalone).

0893-6080/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2010.05.002

MC_Rack by Multi Channel Systems, NeuroExplorer and Offline Sorter by Plexon, Conductor and Mobius by Alphamed), there are still no automated tools suitable to manage and support such a large amount of data and with the possibility to easily extend their functionalities. In other terms, no chance is given to the user to operate, speedily, a multi-channel analysis on more than one recorded stream at a time. Additionally, commercial software tends to have only limited possibilities to incorporate new tools or modify existing ones. For these reasons, new software tools for massive data management and signal processing have to be developed. Recently a number of scientists have started developing custom-made tools capable to analyze multi-electrode recorded data, such as Mea Tools (Egert, Knott et al., 2002), MeaBench (Wagenaar, DeMarse, & Potter, 2005), FIND (Meier, Egert, Aertsen, & Nawrot, 2008), BSMART (Cui, Xu, Bressler, Ding, & Liang, 2008). Unfortunately the cited tools do not provide the users with a large number of algorithms for data analysis and they are not able to manage massive quantities of data at a time. These are the main motivations that induced us to develop a new and innovative software package, named SPYCODE which aims at overcoming such limitations. SPYCODE provides a working environment able to perform efficient data management and processing since it incorporates a very rich repertoire of standard and advanced signal analysis tools. Furthermore, it includes the novel analyses published by our group in recent years (Chiappalone, Vato, Berdondini, Koudelka-Hep, & Martinoia, 2007; Garofalo, Nieus, Massobrio, & Martinoia, 2009; Maccione et al., 2009; Pasquale, Martinoia, & Chiappalone, 2009; Pasquale, Massobrio, Bologna, Chiappalone, & Martinoia, 2008). A few examples of "unconventional" analysis are given by information theory methods, extraction of connectivity maps, self-adapting burst and network burst detection. In the following, we will present the functionalities of our software and an example of application to data recorded from cortical cultures during a neuropharmacological study.

## 2. Methods

In this section we present a concise review of the theoretical background of the signal processing algorithms implemented in SPYCODE.

### 2.1. Spike analysis

#### 2.1.1. Spike detection

Spike timing is the first information to extract from raw data. Since typical signal to noise ratios are much larger than one, the most used method to identify the spikes is a threshold-based algorithm (Maeda, Robinson, & Kawana, 1995) resulting in a point process (e.g. spike train, $ST$) in which each element represents the position in time of a spike.

$$ST(t) = \sum_{s=1}^{N} \delta(t - t_s). \tag{1}$$

Eq. (1) reports the formal definition of a spike train, where $t_s$ is the timing of a spike, $N$ is the number of recognized spikes and $\delta(t)$ is the Kronecker delta function.

After obtaining the spike trains and before proceeding with further analysis, it is important to take into consideration the issue of stationarity, which plays a central role when dealing with neuronal signals. Indeed, experiments performed through non-implantable MEAs (either involving slices or dissociated cultures) generally do not present the non-stationarity typical of in vivo experiments, neither when spontaneous activity is recorded nor stimulation protocols are applied. Hence, the main assumption upon which SPYCODE is based and so data analyzed is the stationarity of recorded neuronal signals, at least for specific time intervals. Furthermore, the possibility to split or join recordings into chunks of desired time length (cf. Section 3.1.1) and the fine tuning of analysis parameters can help detecting possible non-stationary "anomalous" activity periods (e.g. occurring after moving MEAs devices (Wagenaar et al., 2006)) and discard them from the analysis. In order to perform the latter selection, and depending on individual user's needs, also additional Matlab toolbox explicitly dealing with stationarity analysis (e.g. GARCH toolbox, the Mathworks) can be used.

#### 2.1.2. Firing rate

Once spikes have been identified, the easiest and most direct way to characterize the level of activity of a cell is computing its Firing Rate (FR). According to Adrian's definition (Adrian, 1928), the firing rate is the number of spikes in a rather large time window and it can be measured from just one representation of the neural activity, as follows (Eq. (2)):

$$\mathrm{FR} = \frac{\int_0^T \left( \sum\limits_{s=1}^{N} \delta(t - t_s) \right) \mathrm{d}t}{T} = \frac{N}{T} \tag{2}$$

with $T$ representing the duration of the recording and $N$ the number of spikes occurring at time $t_s$. If we count the spikes in a small window of size $\Delta t$, centered at $(t_a - t_b)/2$ and we divide by the bin width, we compute the Instantaneous Firing Rate (IFR) (Rieke, Warland, de Ruyter van Steveninck, & Bialek, 1997).

Dealing with MEAs, tens of channels are likely to be involved during an experimental session. For this reason, it is useful to compute the FR or the IFR of the whole culture and see how it changes given the delivered stimulation. These quantities are simply obtained by computing the FR and the IFR of each single channel and then averaging among all the active electrodes of the MEA, obtaining the Mean Firing Rate (MFR) and the Average Firing Rate (AFR) of the network.

#### 2.1.3. Inter-spike interval histogram

The ISI distribution is the probability density of time intervals between consecutive spikes and it is a useful statistics for describing spike trains (Dayan & Abbott, 2001). The formula to calculate the ISI histogram is reported below (Tam & Gross, 1994):

$$\mathrm{ISI}(\tau) = \sum_{s=1}^{N-1} \delta(t_{s+1} - t_s - \tau) \tag{3}$$

where $\tau$ represents the ISI. As reported by the literature (Perkel, Gerstein, & Moore, 1967), for finite samples of data, such as the observed neuronal spike trains, the ISI histogram (ISIH) serve as an estimator of the actual probability density function. Different shapes of the ISIH give an estimate about the synchronization of the neural network, and each shape denotes a set of timing with shared properties. For this reason, the ISIH can provide a detailed way to classify the dynamic pattern of neurons, e.g. 'spiking' or 'bursting', since bursting neurons usually display "bimodal" ISI histograms (Cocater-Zilgien & Delcomyn, 1992; Tateno, Kawana, & Jimbo, 2002). However, it has been recently demonstrated (Selinger, Kulagina, O'Shaughnessy, Ma, & Pancrazio, 2007) that plotting histograms of logarithmic ISI instead of linear ISI can be useful in better discriminating between intra-burst and inter-burst intervals (Pasquale et al., 2009). For the above reason, the possibility to compute and plot logISIH has been implemented within SPYCODE.

### 2.2. Burst analysis

#### 2.2.1. Burst detection

Highly non-uniform spike timing, or spontaneous bursting, is observed in a wide range of in vitro neuronal preparations and developing organisms (Ben-Ari, 2001). Non-uniformity is present at different time scales (Corner, 2008; Wagenaar et al., 2006) and as a result a variety of algorithms has been proposed for detection of bursts. SPYCODE combines several algorithms to facilitate comparisons and thorough analysis of neural activity time structure.

Generally, a 'burst' consists of a fast sequence of spikes with duration equal to the sum of the inter-spike intervals (ISIs) within the burst itself and separated by an interval relatively long compared to the burst duration (Tam, 2002).

According to this definition, a *burst train* $BT(t)$ of $M$ bursts consists of a sequence of rectangular functions, denoted $\Pi\left(\frac{t-t_m}{\tau_m}\right)$, each of which representing a burst centered at $t_m$ and lasting $\tau_m$,

$$BT(t) = \sum_{m-1}^{M} \Pi\left(\frac{t-t_m}{\tau_m}\right). \tag{4}$$

The symbol $\Pi\left(\frac{t}{\tau}\right)$ is defined as follows,

$$\Pi\left(\frac{t}{\tau}\right) \equiv \begin{cases} 1 & |t| < \tau/2 \\ 0 & |t| > \tau/2 \end{cases} \tag{5}$$

denoting a *rectangular function* with unit amplitude and duration $\tau$ centered at $t = 0$.

### 2.2.2. Network burst detection

When the bursting behavior is organized in array-wide barrages involving the entire network at the same time, the phenomenon is described in the literature with the name of *network burst* (Van Pelt, Wolters, Corner, Rutten, & Ramakers, 2004). Network bursts consist of sequences of synchronized single-channel bursts, which spread across all or part of the MEA (Eytan & Marom, 2006; Raichman, Volman, & Ben-Jacob, 2006; Van Pelt et al., 2004). When observed at a coarse timescale, these bursts seem to be highly synchronized, but they actually hide different spatio-temporal structures of activity propagation, which repeat over time. Moreover, cultures usually exhibit different bursting patterns during the *in vitro* development (Wagenaar et al., 2006). These structures are hypothesized to be correlated to the activation of different functional groups of cells in the network (Baruchi, Volman, Raichman, Shein, & Ben-Jacob, 2008). In order to study the spatio-temporal patterns of activity propagation within network bursts, we devised an algorithm to cluster single-channel bursts in network bursts according to the initial spike's time, named 'burst event' (Cozzi, D'Angelo, & Sanguineti, 2006; Pasquale et al., 2009) (cf. Section 3.1.1).

## 2.3. Post-Stimulus Time Histogram

In order to detect and evaluate the effect of a set of stimuli on the train of spikes, it is common and very useful to compute the Post-Stimulus-Time Histogram (PSTH) (Baljon, Chiappalone, & Martinoia, 2009; Kass, Ventura, & Brown, 2005). Spike trains in response to repeated presentation of the same stimulus are not identical, so the PSTH represents the average behavior of one or more cells to a stimulation pattern. More specifically, the PSTH shows the probability of firing as a function of time after the stimulus onset. This measure is equivalent to a cross correlation between the train of stimuli and the train of spikes. The mathematical definition of the PSTH ($P(t)$) is reported below (Eq. (6)):

$$P(t) = \frac{1}{N}\sum_{i=1}^{N} ST(t - t_{\mathrm{stim}_i}) \tag{6}$$

where $N$ is the total number of stimuli delivered to the network, $t_{\mathrm{stim}_i}$ is the timing at which each stimulus $i$ occurs and ST is the spike train recorded at a specific channel. The time-axis is generally divided in bins of amplitude $\Delta\tau$. If the histogram is normalized by the number of stimulus presentation and also by the bin size (not reported in the above formula), the resulting PSTH gives the instantaneous firing rate – or the probability per unit time of firing, if each bin contains at most 1 spike – as a function of time (Rieke et al., 1997).

Currently, a strong interest is put on the analysis of individual rather than average responses (Quiroga & Panzeri, 2009). Therefore, SPYCODE also allows computing the distribution of response sizes from a series of stimuli and the distribution of the number of spikes per time bin for an individual stimulus.

Stimulus artifacts produced on the recording traces by the electrical stimulation pulses can be easily recognized and accounted by means of the SD algorithm, in which a threshold for artifact detection has to be set by the user. As a result, the Stimulus Artifact Train (SAT) for each recording channel is stored in the following structure:

$$\mathrm{SAT}(t) = \sum_{i=1}^{N} A_{ST_i}\delta(t - t_{\mathrm{stim}_i}) \tag{7}$$

where $A_{ST_i}$ and $t_{\mathrm{stim}_i}$ are the amplitude and timing of the $i$-th artifact. The PSTH is then calculated by means of the Eq. (6). Alternatively, an auxiliary channel containing the digital trigger signal coming from the stimulator (e.g. STG 2004 by MCS) can be recorded as well, providing a way to easily detect the stimulation artifacts and synchronize recording and stimulation.

## 2.4. Cross correlation and information theory

### 2.4.1. Cross-correlogram

Identification of the causal relationships between pairs of neurons is crucial while studying synaptic interactions within the nervous system (Salinas & Sejnowski, 2001). The simplest approach uses the cross correlation function between pairs of spike trains, which measures the probability $C_{xy}(\tau)$ of observing a spike in one train $Y$ at time $(t+\tau)$, given that there was a spike in a second train $X$ at time $t$ (Knox, 1981; Rieke et al., 1997); $\tau$ is called the time-shift or the time lag. When two trains are one and the same (e.g. $C_{xx}(\tau)$), we have the auto-correlation histogram or auto-correlogram.

Given two spike trains (i.e. $x$ and $y$) from two electrodes of a MEA, we count the number of spikes in the $y$ train within a time frame around the spikes of the $x$ train of $\pm T$ (in the order of tens of milliseconds), using bins of amplitude $\Delta\tau$ (usually set at multiple of the sampling frequency). The correct $C_{xy}(\tau)$ is obtained by means of a normalization procedure, by dividing each element of the array by the square root of the product between the number of peaks in the $x$ and the $y$ train, according to the following formula:

$$C_{xy}(\tau) = \frac{1}{\sqrt{N_x N_y}} \sum_{s=1}^{N_x} \sum_{t_i=\left(\tau-\frac{\Delta\tau}{2}\right)}^{\left(\tau+\frac{\Delta\tau}{2}\right)} x(t_s)y(t_s - t_i) \tag{8}$$

where $t_s$ indicates the timing of a spike in the $x$ train, $N_x$ is the total number of spikes in the $x$ train and $N_y$ is the total number of spikes in the $y$ train. In this way, the $C_{xy}(\tau)$ belongs to the interval $[0, +1]$, where the value '1' is generally reached only in the case of an auto-correlation and it corresponds to the value of the peak at zero lag. Moreover, the symmetry between $C_{xy}(\tau)$ and $C_{yx}(\tau)$ is maintained, according to the following relationship: $C_{xy}(\tau) = C_{yx}(-\tau)$. In this way, many of the parameters that we want to extract from the cross-correlogram are symmetric and the computation can be faster (only half of the possible pairs of electrodes needs to be calculated).

To assess the significance of the results, it is often desirable to compute the cross correlation between two channels that is due to chance. In order to meet such needs we introduced in SPYCODE the possibility to shuffle the peak trains obtained after the spike detection. Once data are shuffled, for example with respect to inter-spike-intervals computed within and between spike trains, the cross correlation algorithm can be used to compare shuffled and not shuffled data so to investigate the amount of correlation present by chance in the recorded neuronal activity. We actually developed and presented this shuffling procedure in a previous work from our group (Pasquale et al., 2008).

### 2.4.2. Information theory methods

In order to extract the maps of functional connectivity from a network, we used both cross correlation (CC) analysis and methods derived from information theory (Shannon, 1948). More specifically, we implemented the algorithms for calculating: Mutual Information (MI), Entropy (E) and Transfer Entropy (TE) (Garofalo et al., 2009).

*Mutual Information*. MI (Borst & Theunissen, 1999a, 1999b) is widely used in many different contexts, including Neuroscience. It depends on all higher order moments of the probability distribution and, therefore, MI is sensitive to both linear and nonlinear correlations. MI is evaluated in two different ways depending on the coding mechanism: a *temporal code*, in which also the spikes position is considered, or a simpler *rate code* (e.g. '1110' corresponds to 3 spikes). MI is computed by evaluating joint and single probabilities of the two neurons $(X, Y)$:

$$\mathrm{MI}(X, Y) = \sum_{x\in X}\sum_{y\in Y} p(x, y) \cdot \log_2 \left(\frac{p(x, y)}{p(x) \cdot p(y)}\right) \tag{9}$$

where $x$, $y$ represent a single event (e.g., $x = 2$, $y = 3$ spikes). All probabilities were estimated following the direct method (Borst & Theunissen, 1999b). MI (Eq. (9)) is symmetric with respect to the exchange of the variables $X$ and $Y$. Consequently, it is not suited to recover information on directionality and causality. However, by evaluating the MI of time-delayed time series (e.g. by taking $X$ as the reference spike train and $Y$ as the delayed spike train), we build a MI function of the time-shift, thus providing information both on the synaptic strength and on causality (Chavez, Martinerie, & Le Van Quyen, 2003; Xu, Liu, Liu, & Yang, 1997). The peak of each MI function was used to create the connectivity map, so the highest MI values were associated to the strongest connections. Directionality, instead, was determined, as for the CC function, by evaluating the latency of the peak value (cf. Section 2.4.3).

*Joint-Entropy*. Joint-Entropy (JE) is a novel method, recently presented by our group in the literature (Garofalo et al., 2009). It is a linear measure as the cross correlation (CC), built by considering the cross inter-spike-intervals (cISI) computed across pairs of neurons. Garofalo et al. suggested that JE is sensitive to the activity patterns showed by the neuronal networks and is capable to distinguish the influence of a specific neuron on the activity of another one. For these reasons, despite its simplicity, the JE measure can be considered as a good alternative method to CC. Moreover, it is important to highlight that the algorithm which evaluate the JE is faster than both Mutual Information (MI) and Transfer Entropy (TE) ones which are very heavy from a computational point of view.

JE is calculated according to the following procedure. Considering $X$ as the reference neuron, for each spike of $X$, a subsequent spike of the target neuron $Y$ is considered and a cISI is defined as time difference. If $X$ and $Y$ are strongly connected, the cISI histogram will show a peak at a specific cISI value, and JE will be close to zero. Conversely, when $X$ and $Y$ are not connected or weakly connected, the cISI histogram will be nearly flat and consequently JE will be high. The JE is defined as (Eq. (10)):

$$\mathrm{JE}(X, Y) = -\sum_{k=1}^{n} p(\mathrm{cISI}_k) \cdot \log_2(p(\mathrm{cISI}_k)) \tag{10}$$

where $p(\mathrm{cISI}_k)$ is the estimated probability of $\mathrm{cISI}_k$.

*Transfer Entropy*. TE is an asymmetric quantity which allows to extract causal relationships from time series (Lungarella, Pitti, & Kuniyoshi, 2007). It shares some of the desired properties of the MI taking also into account the history and the linear as well as nonlinear causal interactions among the peak trains (Gourévitch & Eggermont, 2007). If we indicate with $x_t$ the number of spikes (of the spike train $X$) falling in the time window $t$, $x_t^m = (x_t, x_{t-1}, x_{t-2}, \ldots, x_{t-m+1})$ represents the spike count vector of the past $m$ time windows. Considering a second spike train $Y$ and its spike count vector $y_t^m$, TE can be defined as (Eq. (10)):

$$\mathrm{TE}_{y\to x} = \sum_{x_{t+1}, x_t^k, y_t^l}^{n} p(x_{t+1}, x_t^k, y_t^l) \cdot \log\left(\frac{p(x_{t+1}|x_t^k, y_t^l)}{p(x_{t+1}|x_t^k)}\right) \tag{11}$$

where $p$ denotes the transition probabilities conditioned to the past $k$ and $l$ observations of the spike trains $X$ and $Y$, respectively. Since TE measures the gain in information of knowing the future and the past of $x_t$, once $y_t^l$ is known, high TE values indicate that the spike train $Y$ influences the response of $X$.

### 2.4.3. Connectivity maps

Connectivity maps can be obtained starting from each of the methods presented above. Since CC and MI function yield a connectivity time series for each couple of neurons, the peak of each function ($C_{\mathrm{peak}}$) was used to create the connectivity matrix: the highest values were associated to the strongest connections. Directionality, instead, was determined by evaluating the latency of the peak value. On the other hand, JE, providing single asymmetric values among each pair of neurons, is directly used to infer causality. However, differently from CC and MI, the strongest connections should be associated to the lowest JE values.

To obtain a connectivity map, a procedure to select the strongest (i.e., statistically significant) connections is necessary since a CC (or MI or TE or JE) value between an electrode pair may be present whether there is a direct or indirect (i.e., multi-hop) link between the two or whether there is a true connection or simply random (i.e. due to noise) co-activations.

The value used to select the strongest connections (threshold the Connectivity Matrix) is a crucial parameter for a correct estimation of the functional links. Driven by heuristic considerations (by looking at the $C_{\mathrm{peak}}$ histogram), we decided to take into account the $K$ strongest links among the possible $N$ ($K < N$). Any threshold (TH) choice, in a sense, is arbitrary since a well-accepted procedure which discriminates the relevant from the trivial $C_{\mathrm{peak}}$ does not exist. In order to infer a reference value which the selected threshold (TH) can be compared to, we calculated the correlations between all electrodes after shuffling the ISIs (but maintaining their original ISI distribution). The shuffling procedure is aimed at destroying any causal relationship between the two spike trains and it represents an estimate of the value (e.g., $C_{\mathrm{peak}}$) expected between two random spike trains having the same ISI distribution and the same firing rates of the original ones. Particularly, we applied the shuffling procedure to the spike trains yielding the strongest 10 $C_{\mathrm{peak}}$ and for each of them the corresponding spike trains $(X_i, Y_i)$ were shuffled 5000 times. Eventually, a shuffled CC threshold ($\mathrm{TH}_{sh}$) was defined as $m + 3$std, where $m$ is the mean and std the standard deviation of the shuffled values. We verified that the selected TH is more restrictive than the $\mathrm{TH}_{sh}$. Maps were created on the basis of the more selective threshold (usually TH).

## 2.5. Avalanche detection and analysis

Periods of synchronized electrophysiological activity are present in different *in vitro* models of the neocortex (Baker, Corner, & van Pelt, 2006; Chiappalone et al., 2007; Raichman & Ben-Jacob, 2008). It has been demonstrated that these events hide a more sophisticated embedded form of dynamics, the so-called 'neuronal avalanches' (Beggs & Plenz, 2003, 2004; Pasquale et al., 2008), whose presence has been often associated to the capability of the network to enhance the information transmission (Plenz & Thiagarajan, 2007).

Starting from the work of Beggs and Plenz (2003, 2004), a neuronal avalanche can be defined as an event of widespread spontaneous electrical activity over the MEA, preceded and followed by a silent period. Recordings can be divided into time windows of duration $\Delta t$ (called bins); inside each bin the spatial distribution of activity over the MEA represents a 'frame'. A frame which does not contain any spike is called a 'blank frame'. An electrode is active in a time bin $\Delta t$ if it records at least one spike inside that time window. Thus, a frame is active if it recruits at least one active electrode. Following these definitions, a neuronal avalanche is a continuous sequence of active frames, preceded and followed by at least one blank frame.

Consequently, the 'avalanche size' can be defined either as the total number of active electrodes within an avalanche, taking into account multiple activations of the same electrodes (definition 1), or as the number of electrodes being active at least once inside an avalanche (definition 2). The duration of an avalanche is usually called 'avalanche lifetime' and is expressed in number of bins $\Delta t$. From the spike-detected signals, we can extract neuronal avalanches and derive the relative histogram of avalanche sizes (following both definitions) and lifetimes. These histograms are generally represented in bilogarithmic scale in order to show whether the distribution follows a power law: thus, if the probability of observing an avalanche of size $s$ is expressed by $P(s) = as^b$, that appears as a linear relationship in the bilogarithmic scale with slope $b$.

## 3. Results

As already underlined in the Introduction, the use of MEAs for electrophysiological experiments is likely to produce a huge quantity of data, thus requiring a smart software which could automatically search for the experiments to be analyzed, recognize them and apply the needed analysis, several times, to different sets of data. We called such an approach 'multiple analysis', indicating the automatic repetition of the same 'multiple' computational operations to 'multiple' data and without the intervention of the user (except for the initial setting of the parameters needed to retrieve the experiments).

The implementation of the 'multiple analysis' is the main feature of SPYCODE, whose structure is described in the following paragraphs. Together with the presentation of the software and in order to point out its functionality and usefulness, we will show, as an example, the results obtained for an experimental session of neuropharmacology.

### 3.1. Software structure

SPYCODE was written in MATLAB 6.5 and it is compatible with all the following Matlab releases (up to R2008b). Although we have tested it exclusively on Microsoft (XP, Vista) and Linux (Ubuntu Jaunty Jackalope 9.04), the software is expected to be compatible with other operating systems within the framework of MATLAB, with a limitation on the conversion part for which the libraries are available only for the Windows platform (see www.multichannelsystems.com for reference).

#### 3.1.1. Main functional blocks

Fig. 1 schematically shows the organization of the software, which includes a block for data conversion, one for data filtering (optional) and the cascade of analysis that can be done once that spike detection has been performed: spike analysis, burst detection and analysis, PSTH, cross correlation, and a set of additional tools with information theory and avalanche detection and analysis computation. The description of each block is briefly reported in the following.

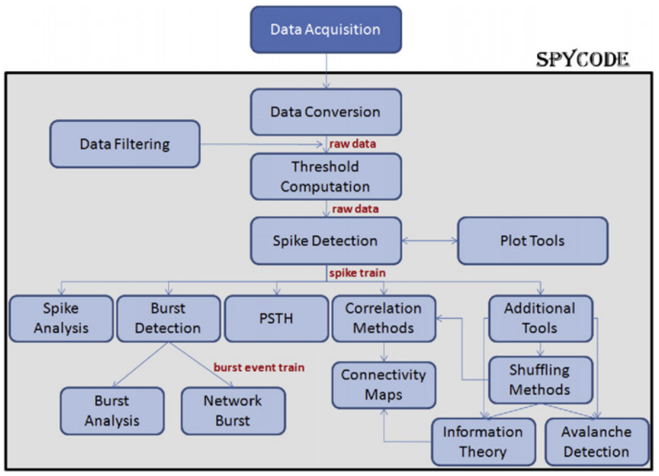


**Fig. 1.** Schematic overview of the data analysis procedure implemented within SPYCODE. The functional blocks show the main computational and algorithmic options offered by the software. The data flow starts with the data conversion procedure, after which raw data are available in Matlab internal format (i.e. *.mat), then a filter can be applied to de-noise the data (optional). The spike detection procedure allows to reduce of a factor of 190 the size of the data and prepares them for a sequence of possible operations.

*Data conversion*. Up to now, the supported data formats are the Multi Channel Systems (Reutlingen, Germany) and the Panasonic MED64 system (Osaka, Japan). We expect to make SPYCODE compatible with other acquisition formats in future releases, as well as adding loading engines for alternative Matlab formats. Data are converted to the internal format where single channels are stored in single files. A serial conversion of individual channel raw data is needed because of Matlab system memory management. All analysis tools then have an output function that combines the results for all channels and all dependent variables into a single file per experimental phase.

*Data filtering*. Given the possible need of filtering the data for noise reduction or frequency study, SPYCODE gives the opportunity to filter the raw data (after the conversion step) through either a low-pass or high-pass filter (Butterworth filtering implemented). The opportunity to choose the appropriate cutoff frequency is given.

*Threshold computation*. The threshold used for the spike detection procedure (cf. Section 2.1.1) is calculated as a multiple of the standard deviation of the raw data noise. In order to do that, the user must appropriately select a period of the recording in which no spike or burst are present (Jimbo, Tateno, & Robinson, 1999). To allow this operation, a custom GUI (*compute_basalthr()*) was built through which the user can visualize the traces of every channel. Given the possible differences between the noise level of channels belonging to the same MEA, single channels can be explored individually and the chunks of recordings to be used can be selected independently. The function *compute_basalthresh()* builds the vector of thresholds used by the spike detection algorithm and containing the data for individual channels.

*Spike detection*. Within SPYCODE, two different algorithms are implemented to detect spikes, both based on the computation of a differential threshold (Bove, Grattarola, & Verreschi, 1997; Maccione et al., 2009; Vato et al., 2004). To the best of our knowledge, SPYCODE is the first to make available spike detection *not* based on a simple hard threshold.

1. The SDDT (Spike Detection Differential Threshold) is a fast algorithm that split up the signal in consecutive windows fitted to hold at most one single spike and assign a spike whenever the absolute Max–Min difference within a window overtakes the threshold.

2. A second algorithm called PTSD (Precision Timing Spike Detection), addresses the need of improving the detection both in terms of timing precision and false positive/negative reduction at the cost of a slightly increased execution time. The PTSD basically looks for all Relative Maximum/Minimum (RMM) within the signal and evaluates spikes firstly by comparing RMMs position in terms of spike expected duration and refractory period and finally by excluding under-threshold outputs.

After the threshold computation, the Spike Detection algorithm can be chosen and the parameters to be used (e.g., refractory period, sliding window length) inserted through a custom GUI. Once the parameters known, and depending on the choice of the user, the functions *peak_detection_sddt()* and *peak_detection_ptsd()* will perform the detection of the spike trains.

As an additional tool, the functions *joinRecPhases_main()* and *splitRecPhases()* give the user the possibility to split spike trains into arbitrarily long chunks and similarly to join different spike trains into a single one. This allows the user to perform analysis on very long experiments (whose raw data had been necessarily split because of PC's resources management) with no need of combining results from different chunks *a posteriori*.

*Plot tools*. The opportunity to plot both the raw data and the raw data together with the already detected peaks is given to the user (*plot_multiplerawdata()* and *plot_singlerawdata()* function respectively). Furthermore, a raster plot of all the channels can be plotted (after choosing the temporal window to display) through the function *plotraster()*. The plots are saved in .jpg or .fig format. Nonetheless, in order to provide a wider portability the possibility to save the plot in the pdf format is given through the function *plotraster_pdf()*.

*Spike Analysis*. The Spike Analysis allows the user to:

1. Compute the Mean Firing Rate (i.e., #spikes/s) of each channel, and choose the minimum rate for it to be considered active (*MAIN_mfr()* function).
2. Compute the Average Firing Rate of each individual channel and choose both the time bin to use and the minimum rate threshold. AFR plots are generated and saved in the .fig and .jpg formats (*AverageFiringRate()* function).
3. Compute the Inter-spike interval (ISI) of each channel separately, choose the parameters for the computation and plot the distribution of all the ISI histograms in a single plot (*ISI_main()* function). All plots are saved in both .fig and .jpg formats.
4. Compute and plot the logarithmic ISIH (logISIH), choosing the desired number of bins per decade on the *x*-axis: optionally the user can plot the logISIH of all channels in a single plot and decide to compute and store the ISI value that best separates intra-burst and inter-burst ISIs (ISIth), needed for the subsequent burst detection analysis.

*Burst Detection and Analysis*. The Burst Detection functional block allows searching for bursts in the recorded activity by following two different approaches.

1. Burst Detection—v1. According to the method presented in the literature few years ago (Chiappalone et al., 2005), the function *BurstDetection()* allows to detect bursts as sequences of at least $N$ consecutive spikes (default value $N = 5$) spaced less than a convenient time threshold (default value 100 ms).
2. Burst Detection—v2. Starting from the approach described above (v1), we developed a new and more reliable algorithm capable of detecting bursts and self-adapting to different experimental conditions (Pasquale et al., 2009). Basically, the new algorithm, implemented in the function *burstdetection_main()*, evaluates the logISIH in order to look for the best threshold (ISIth) between intra-burst (i.e. within bursts) and inter-burst (i.e. between bursts and/or outside bursts) activity. If ISIth is lower than 100 ms, the new BD uses the ISIth value to determine the maximum ISI allowed within a burst. On the contrary, if ISIth is higher than 100 ms, the new BD uses two different thresholds: the first one (i.e. 100 ms) is used for detecting burst cores, whereas the second one (i.e. ISIth) is used to extend burst cores at the boundaries to include all spikes whose ISI is lower than ISIth.

Once the burst detection has been performed, either following v1 or v2, the user can compute and save in .txt and .mat files the main statistics related to the burst analysis (e.g., number of bursts, mean bursting rate, mean frequency intra-burst, burst duration, etc.) by calling the functions *MAIN_StatisticReport()* or *MAIN_StatisticReportMean()*.

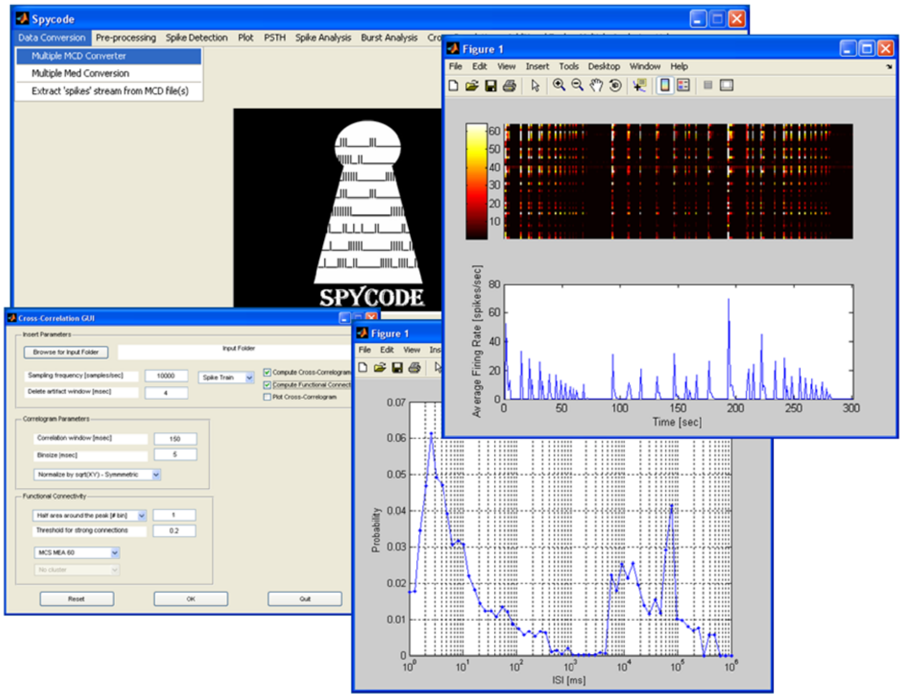
*Network burst detection and analysis*. SPYCODE offers the user the possibility to detect network bursts from the burst trains (i.e. the output of the burst detection procedure). In our algorithm (*netBurstDetection_main()*), we consider the *cumulative burst event train*, namely the sequence of all the initial spikes of each single-channel burst in the array and we compute the logarithmic inter-burst event interval histogram (logIBeIH) by binning data in equally spaced logarithmic bins (*IBEI_main()*). In this way, we are able to highlight the presence of two or more distinct peaks, the first one corresponding to short intervals within network bursts and the others to long intervals between network bursts. Hence, the same approach developed for the detection of spike bursts is applied to the NBD, starting from the cumulative burst event train: the only operation needed is setting two parameters, firstly the maximum inter-burst event interval for burst events within a network burst (maxIBeI) and secondly the minimum percentage of recording electrodes involved in a network burst (usually set at 20% the total number of active channels). The maxIBeI is set according to the IBeI threshold extracted from the logIBeIH, while the minimum percentage of recording electrodes is set by the user.

*PSTH*. The PSTH functional block allows computing the response to the stimuli (PSTH area and latency) of all channels by inserting the appropriate parameters for artifact blanking. The Matlab function we used is *MainPSTHarea()* function, which computes the number of spikes in the considered window and once the blanking given to artifact has been performed. It also allows to plot the histograms of single channels individually or together with all the others (i.e., a multiple plot in which the PSTH of all the electrodes are plotted together). The functions used are, respectively, *plot_singlepsth()* and *plot_multiplepsth()*. All the generated plots are saved in both the .fig and .jpg formats.

*Correlation methods and connectivity maps*. A dedicated GUI for cross correlation computation and analysis was developed in SPYCODE. Through the GUI, the user can select the time bin and the temporal window of the correlogram, together with different methods for the normalization of the correlogram itself. The possibility to define a threshold for the 'strong' connections is given (cf. Section 2.4.3) in order to plot the connectivity maps also starting from the CC algorithm. The correlation can be calculated both in spike trains (i.e. the output of the spike detection procedure) and in burst event trains (i.e. the output of the burst detection algorithm).

*Information theory*. Entropy, Joint-Entropy, Transfer Entropy, and Mutual Information can be evaluated by launching the *Information Theory GUI* which allows the user to choose between a *Rate* and a *Temporal* approach (cf. Section 2.4.2). Parameters (as the bin width and the window size) have to be set by the user. Maps are obtained by selecting several threshold values.

*Data shuffling*. As already introduced (cf. Section 2.4.1), an additional tool allowing to shuffle the peak trains obtained after detection is provided. The method was already presented in a previous paper from our group (Pasquale et al., 2008) but now it has been integrated in the SPYCODE framework and made available to the



Cross-Correlation GUI

Insert Parameters

Browse for Input Folder | Input Folder

Sampling frequency [samples/sec]: 10000

Delete artifact window [msec]: 4

Spike Train

[✓] Compute Cross-Correlogram

[✓] Compute Functional Connect

[ ] Plot Cross-Correlogram

Correlogram Parameters

Correlation window [msec]: 150

Binsize [msec]: 5

Normalize by sqrt(XY) - Symmmetric

Functional Connectivity

Half area around the peak [# bin]: 1

Threshold for strong connections: 0.2

MCS MEA 60

No cluster

Reset | OK | Quit

**Fig. 2.** Running SPYCODE. The main Graphical User Interface (GUI) is depicted during a working session of the software. Additional GUIs for data input are shown to the user before launching each computation. The output of an analysis session is provided through a series of figures (.fig or .jpg format) or results files (.mat or ASCII format).

user. Briefly, the method allows binning the spiking activity into appropriately wide temporal windows and then permuting the order of the bins either separately for each channel or for all channels at the same time. This procedure can be useful, for example, to validate the results obtained from the avalanche analysis or for calculating the excess amount of spike correlation.

*Neuronal avalanches*. A dedicated GUI allows the user to launch the neuronal avalanches detection and analysis (*aVaGUI()*): the user can set the time resolution for detecting avalanches and decide whether apply a logarithmic binning. The same GUI offers the possibility to apply the neuronal avalanches analysis to other MEA layouts and visualize the results' figures.

### 3.1.2. The Graphical User Interface and the menu bar

The SPYCODE software package has been developed through a user-friendly Graphical User Interface (GUI) in which also a non-experienced MATLAB user is able to perform data analysis without knowing the details of the source code. Fig. 2 shows a screen capture of a SPYCODE user session with the GUI running under Windows.

The GUI gives access to a comprehensive menu bar (see Fig. 3 for a detailed structure) through which the user can choose the analysis to perform. This choice was taken to allow the developers to add new functionalities to the software by simply adding new menu or submenus to the main interface. Hence, SPYCODE is a dynamic tool which can be easily modified and integrated with additional algorithms, though keeping its core structure unchanged. The GUI menu is split into sections oriented by function type. The first level consists of eleven menus, namely, 'Data Conversion', 'Pre-processing', 'Spike Detection', Plot', 'PSTH', 'Spike Analysis', 'Burst Analysis', 'Cross Correlation', 'Additional Tools', 'Multiple Analysis' and 'Help'. This last one contains a brief documentation on how to use the software.

The 'Data Conversion' menu is responsible for data format conversion starting from files recorded by MCS or MED64 system (up to now). The 'Pre-processing' menu opens a GUI for selecting the desired filtering option to be applied to the raw data in .mat format. The threshold computation and the shortcuts to two different spike detection procedures are available under the 'Spike Detection' voice. The 'Plot' menu offers the possibility to plot raw data and raster plot, with the option to add the Average Firing Rate (AFR) profile to the raster. The 'PSTH' contains all the voices for calculating the histogram profile, the latencies of the first evoked spikes (for each channel), the area of the histogram and the plot options, including the raster plot in a defined window after the stimulation (i.e. stim raster). In the 'Spike Analysis' menu the three functions for analyzing the spike train are provided, while in the 'Burst Analysis' both the two detection methods (i.e. burst detection v1 and v2) and the analysis tools are implemented. The 'Burst Detection v2' voice offers three additional submenus for the actual detection of bursts by following the new approach (cf. Section 3.1.1), the Inter-Burst event Interval (i.e. 'IBeI) analysis and the Network Burst Detection computation, starting from the burst events. The 'Cross Correlation' voice is responsible for the correlogram computation and the plot tools to visualize the computed histograms. The 'Additional Tools' contains the shortcuts to the GUI for the Avalanche Detection and Analysis, the Information Theory Methods and the Shuffling Methods. Finally, the last voice 'Multiple Analysis' is responsible of the GUI for selecting the experiments on which the user wants to performs the same set of analysis (see Section 3.2.1).

## 3.2. Data management and multi-channel data processing

### 3.2.1. The 'multiple analysis' philosophy

The 'multiple analysis' approach implemented within SPYCODE relies upon the directory tree structure generated by the software itself. The latter is engineered so to create different folders for the different analysis performed. The two most important folders are the ones containing raw data in .mat format and the one containing the spike trains, built after the spike detection was performed. Given the directory tree generated by the program and the common practice of archiving different experiments with different

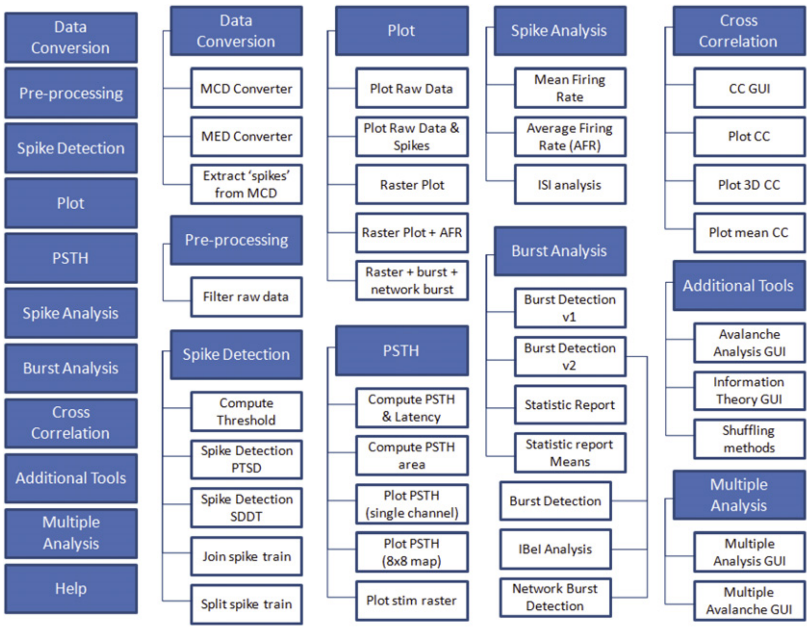
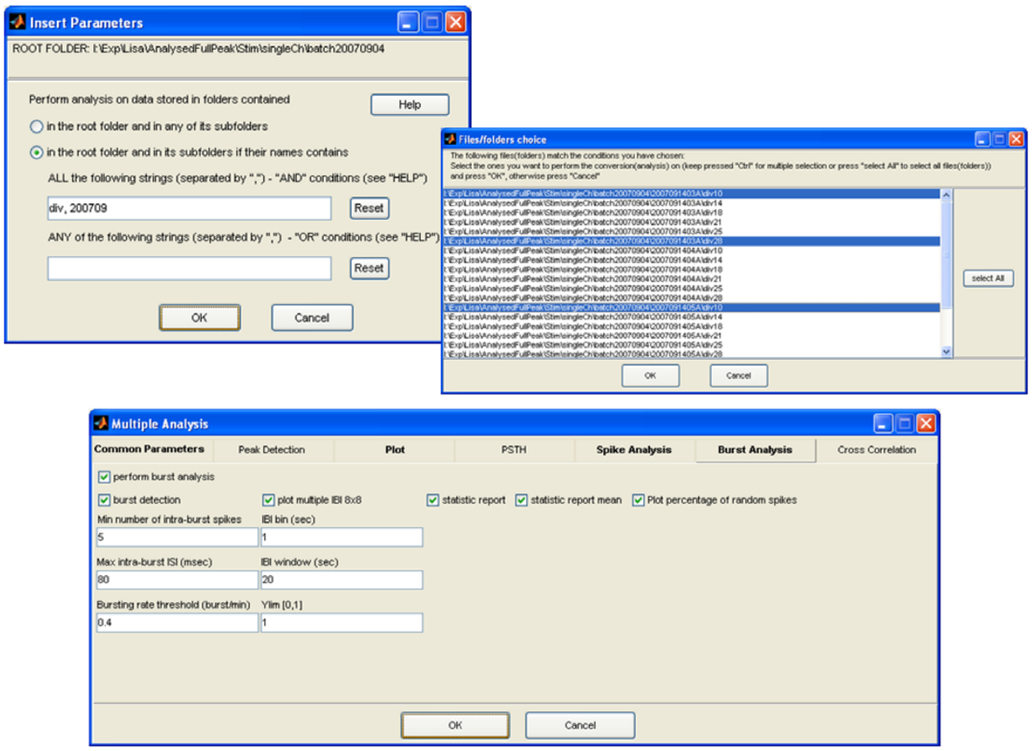


**Fig. 3.** The complete structure of the SPYCODE menu. Each voice contains a set of submenus that allow the user to perform several tasks related to a specific operation.


Insert Parameters

ROOT FOLDER: I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904

Perform analysis on data stored in folders contained

○ in the root folder and in any of its subfolders

◉ in the root folder and in its subfolders if their names contains

ALL the following strings (separated by ",") - "AND" conditions (see "HELP")

div, 200709

ANY of the following strings (separated by ",") - "OR" conditions (see "HELP")

Files/folders choice

The following files(folders) match the conditions you have chosen:
Select the ones you want to perform the conversion(analysis) on (keep pressed "Ctrl" for multiple selection or press "select All" to select all files(folders))
and press "OK", otherwise press "Cancel"

```
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091403A\div10
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091403A\div14
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091403A\div18
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091403A\div21
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091403A\div25
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091403A\div28
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091404A\div10
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091404A\div14
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091404A\div18
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091404A\div21
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091404A\div25
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091404A\div28
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091405A\div10
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091405A\div14
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091405A\div18
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091405A\div21
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091405A\div25
I:\Exp\Lisa\AnalysedFullPeak\Stim\singleCh\batch20070904\2007091405A\div28
```

Multiple Analysis

Common Parameters | Peak Detection | Plot | PSTH | Spike Analysis | Burst Analysis | Cross Correlation

[✓] perform burst analysis

[✓] burst detection [✓] plot multiple IBI 8x8 [✓] statistic report [✓] statistic report mean [✓] Plot percentage of random spikes

| | |
|---|---|
| Min number of intra-burst spikes: 5 | IBI bin (sec): 1 |
| Max intra-burst ISI (msec): 80 | IBI window (sec): 20 |
| Bursting rate threshold (burst/min): 0.4 | Ylim [0,1]: 1 |

**Fig. 4.** Multiple analysis GUIs. The user can choose the experiments to analyze through a window which allows to insert the substrings the names of the experiments must contain (top). Once all the experiments have been retrieved, the user can perform a final check and select the ones the analysis will be applied to (middle). Finally, the user is prompted with a comprehensive window through which he can choose the analysis to perform and the parameters to be used. The analysis selected will be applied on the chosen experiments upon confirmation and without any further intervention of the user (bottom).

names and in different folders, the philosophy underlying the multiple analysis, can be outlined with the following steps:

(1) Choose the root folder from which to start the search for the experiments to analyze.
(2) Give the sequences of characters that the name of the experiment/s to be analyzed must fulfill.
(3) Choose the analysis to perform on the experiments and the parameters to be used.
(4) Launch the analysis.

The key point in the previous sequence of steps is the second. Indeed, it allows retrieving any folder, experiment or raw data file by exploring the entire directory tree starting from the given root folder. By using the MATLAB scripts, we built *ad hoc* GUIs through which the user can search the needed experiment folder and perform the desired analysis (Fig. 4). The underlying search policy is that of 'regular expression'.

A regular expression is a sequence of characters through which a set of strings can be identified. It is made up of normal and special characters (i.e., characters with special meaning) and it is written in a formal language. The regular expression principle underlies the "multiple analysis" implementation procedures but it is completely transparent to the user. The latter in simply required to insert the substrings contained in the name of the experiments he wants to analyze (considering its entire path).

The choice of the aforementioned substrings, which the name of the searched experiments must fulfill, is done in either an AND or OR modality, in order to supply the user with a wide range of research criteria (see Fig. 4, top). Once the final list of experiments is extracted, a custom window (see Fig. 4, middle) displays it and allows the user to choose the final set of experiments on which to perform the analysis. As the last step, a window through which the user can choose the analysis to perform and the parameters to be used and finally launch the analysis is prompted (see Fig. 4, bottom).

As a final check, once the list is complete and the analysis started, the software will check the presence in the experiments folders of the data needed to perform the requested analysis. If such data (i.e., folders) are not present, the current analysis is skipped and the following is performed. For example, let us suppose the user has done only the peak detection analysis on the raw data of an experiment and he now wants to perform the spike analysis and the cross correlation on spikes and bursts. He gives SPYCODE the instructions to perform the analysis desired and launch the application. The software performs the spike analysis on the data and then it computes the cross correlation on spike trains. But, when it has to perform the cross correlation on the bursts, it finds no Burst Analysis folder and it simply skips the command by passing at the next experiment.

### *3.2.2. The use of 'MEX' files*

MATLAB is built on many well-optimized libraries for creating and managing vectors, matrices and multi-dimensional arrays. Nevertheless in some situations it can be useful to run instruction outside the MATLAB environment in order to speed the execution time. MEX files serve this purpose and are mainly used for two reasons: bypass bottleneck computations like for-loops or invoke external routines like multithreaded libraries. The former scenario happens anytime it is not possible to convert iterative instructions into equivalent array-based operations, i.e. vectorizing algorithms to take advantage of MATLAB's kernel implementation. In such a case, the problem can be overcome by calling compiled C/FORTRAN functions using MATLAB's MEX files. A straightforward C implementation of the cross correlation algorithm can speed up the code by a factor of $>20$ over an optimized MATLAB native language program. Similarly, spike detection can be time optimized by a factor $>60$ allowing to process 60 channels recordings (acquired at 25 kHz) at 0.4 times real time on an Intel Pentium 4 3.4 GHz with 4 GB working memory. This allows our software to work both for overnight 'number crunching' on a large data set, and on-the-fly for exploratory data analysis.

## *3.3. Application to neuropharmacological investigations*

In what follows we illustrate the utility of SPYCODE, with respect to both the wide range of analysis it allows to perform and the possibility to batch process huge amount of data. We employ a small sample dataset from cortical networks coupled to MEAs. A brief description of the preparation and employed data set is given.

### *3.3.1. Biological preparation and experimental set-up*

Primary cortical cultures were obtained from brain tissue of Sprague Dawley rats at embryonic day 18 (E18) using protocols reported in previous works (Chiappalone et al., 2008; Pasquale et al., 2008). The experimental set-up was based on the MEA60 system (Multi Channel Systems, Reutlingen, Germany) with a gain set to 1200; a PC equipped with the MC Card, a PCI A/D—D/A board with a maximum of 128 recording channels, 12 bits resolution. Raw data were acquired at a sampling frequency of 10 kHz/channel using the commercial software MCRack (Multi Channel Systems). The experiments of chemical manipulation started with 10 min recording of the spontaneous activity in physiological solution (NaCl 150 mM, KCl 2.8 mM, $CaCl_2$ 1.3 mM, $MgCl_2$ 0.7 mM, Hepes 10 mM, glucose 10 mM, pH 7.3), followed by 10 min recording under drug treatment and additional 10 min recording after drug wash-out, again in physiological solution. More specifically, we tested the effects of bath applied bicuculline (BIC, antagonist of the inhibitory pathways mediated by the receptor $GABA_a$ – final concentration 30 μM) (Gramowski et al., 2004) and D-2-amino-5-phosphonopentanoic acid (APV, antagonist of the NMDA glutamatergic receptors – final concentration 25 μM) (Gross et al., 1995). All agents were purchased from Sigma Aldrich (St. Louis, MO, USA).

### *3.3.2. SPYCODE to analyze neuropharmacological data*

By following the procedure explained in Fig. 1 (data analysis diagram), we first converted our data, then calculated the threshold and applied the PTSD algorithm to detect spikes. Given the spike trains of our experiments, by using the 'Multiple Analysis' interface, we computed both raster plot and cross correlation on spike trains. We then calculated the Joint-Entropy in order to extract the functional connectivity maps of our cultures during the three experimental conditions.

As we can infer from the analysis performed with SPYCODE, the application of BIC and APV produces a significant change in the behavior of the tested cortical networks. The raster panels in Fig. 5 show the electrophysiological activity in two different experiments: a network under exposure of BIC (top panels, Fig. 5A, B, C) and a network under exposure of APV (bottom panels, Fig. 5D, E, F). During spontaneous activity (i.e. no drug addition, Fig. 5A and D) the behavior of the two networks is similar, even if more sustained in the bottom experiment, and presents the typical burst pattern characterized also by the presence of the 'network bursts' (cf. Section 2.2.2). When a chemical is added, the behavior completely changes in a quite opposite direction. Under the effect of 30 μM BIC (Fig. 5B), the activity appears organized in larger bursts, highly synchronized and involving almost all the recording channels. During the application of 25 μM APV (Fig. 5E) to the culturing medium the level of activity abruptly decreases in terms of burst pattern but the network appears still synchronized. After the medium change, the BIC effect does not disappear (Fig. 5C), even if less bursts are generated, while in the APV experiment (Fig. 5F) the wash-out allows the return to the initial condition after a first assessment phase of about 120 s.

Similar conclusion can be drawn by looking at the cross correlation histograms from spike trains depicted in Fig. 6. The correlograms for one representative channel versus all the others are reported in the three experimental conditions for the two experiments (Fig. 6: top, BIC experiment; bottom, APV experiment). In both cases, the presence of the drug increases the synchronization level among the channels with respect to the basal condition. After the wash-out, the return to the initial status is reached only in the case of an APV experiment and not in case of BIC. This conclusion is in accordance with previous results reported in the literature (Arnold et al., 2005; Bonzano, Bove, & Martinoia, 2006).

Maps reported in Fig. 7 top, represent the functional connections obtained by applying Transfer Entropy to the BIC experiment. The strongest sixty connections are plotted for each experimental phase, suggesting that the topological organization of the most correlated channels clearly changes from the spontaneous to the BIC-stimulated condition (Fig. 7A and B). The effect of BIC is maintained also after the wash-out, without significantly changing the

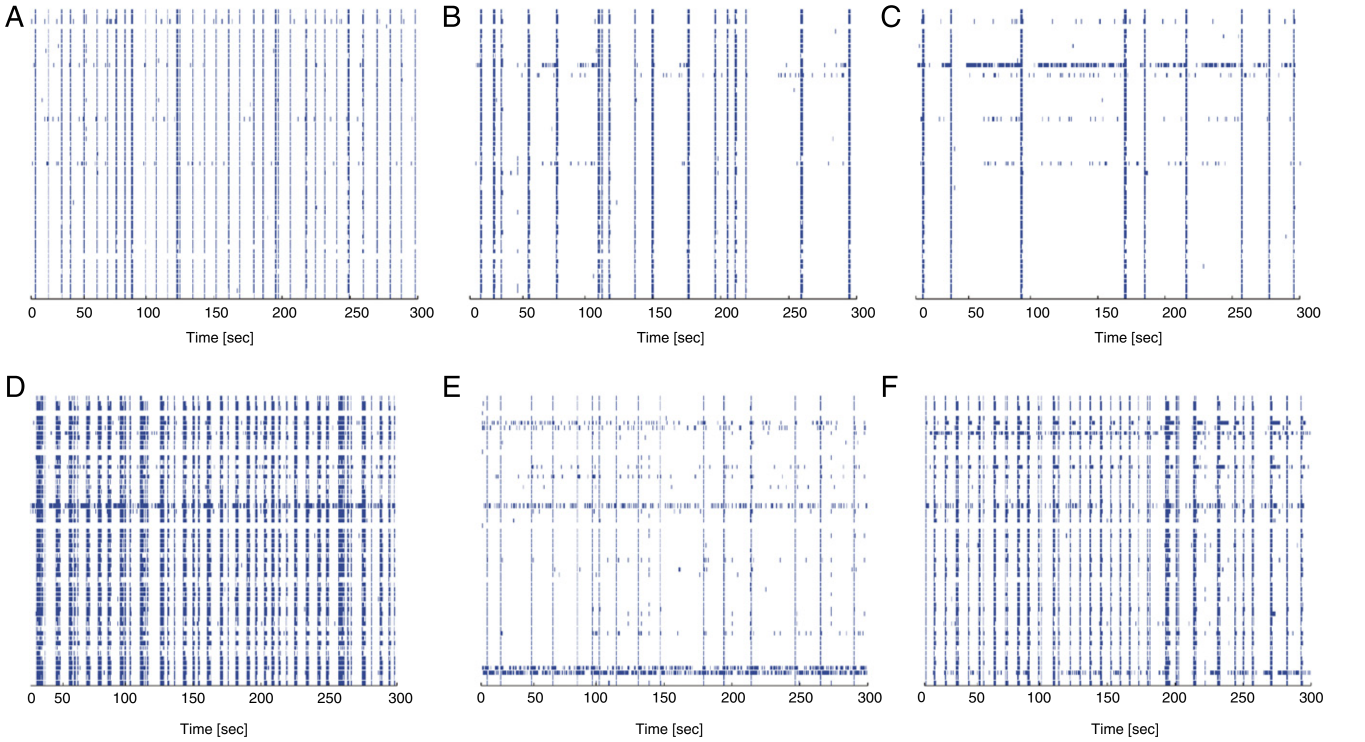


**Fig. 5.** Raster Plot of 60 recording channels from two MEAs during two experiments of chemical manipulation: top, BIC 30 μM; bottom, APV 25 μM. A, C, D, F: spontaneous activity. B: BIC exposure (30 μM). E: APV exposure (25 μM). Time scale 0–300 s.

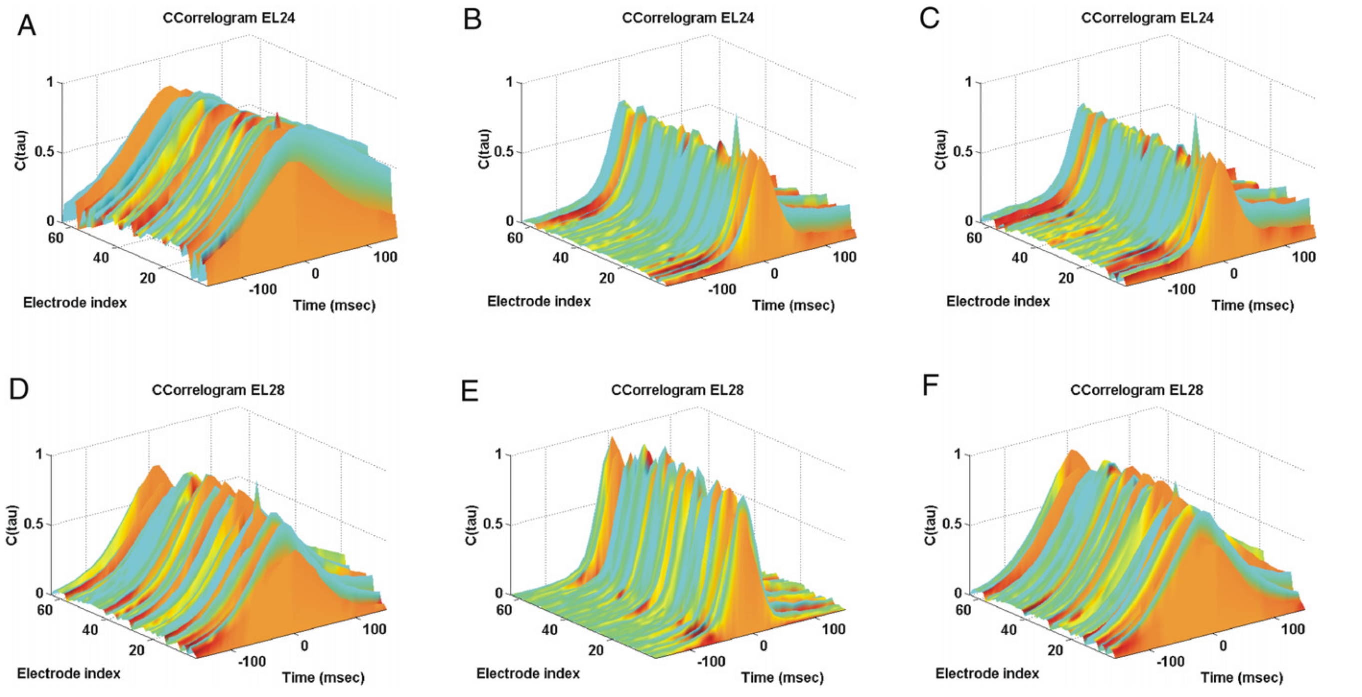


**Fig. 6.** Cross correlations during two experiments of pharmacological manipulation (top, BIC; bottom, APV). 3D cross-correlograms between one channel (electrode 24, top and electrode 28, bottom) and all the others during the initial spontaneous activity (A, D), the chemical treatment (BIC 30 μM, B and APV 25 μM, E) and again spontaneous activity after wash-out (C, F). Bin size is 10 ms, window length is [−150, +150] ms.

distribution of the connections (Fig. 7C). In the case of APV experiment (Fig. 7, bottom) the maps do not qualitative changes during the three phases, since the involved channels remain roughly the same.

Besides the scientific results we reported, the usefulness of SPYCODE can be inferred even at this very simple analysis level by observing that, for example, an accurate spike detection together with a smart visualization tool allow the user to have an immediate visual feedback of the cultures' behavior for several experiments. Most interestingly, the parameters used for the analysis (e.g. raster plot time window) must be set once (during the parameter choice phase) and no operation has to be repeated even if the number of experiments to be analyzed is huge (cf. Section 3.2.1). As in the case represented in Figs. 5–7, the user is prompted with all results, for

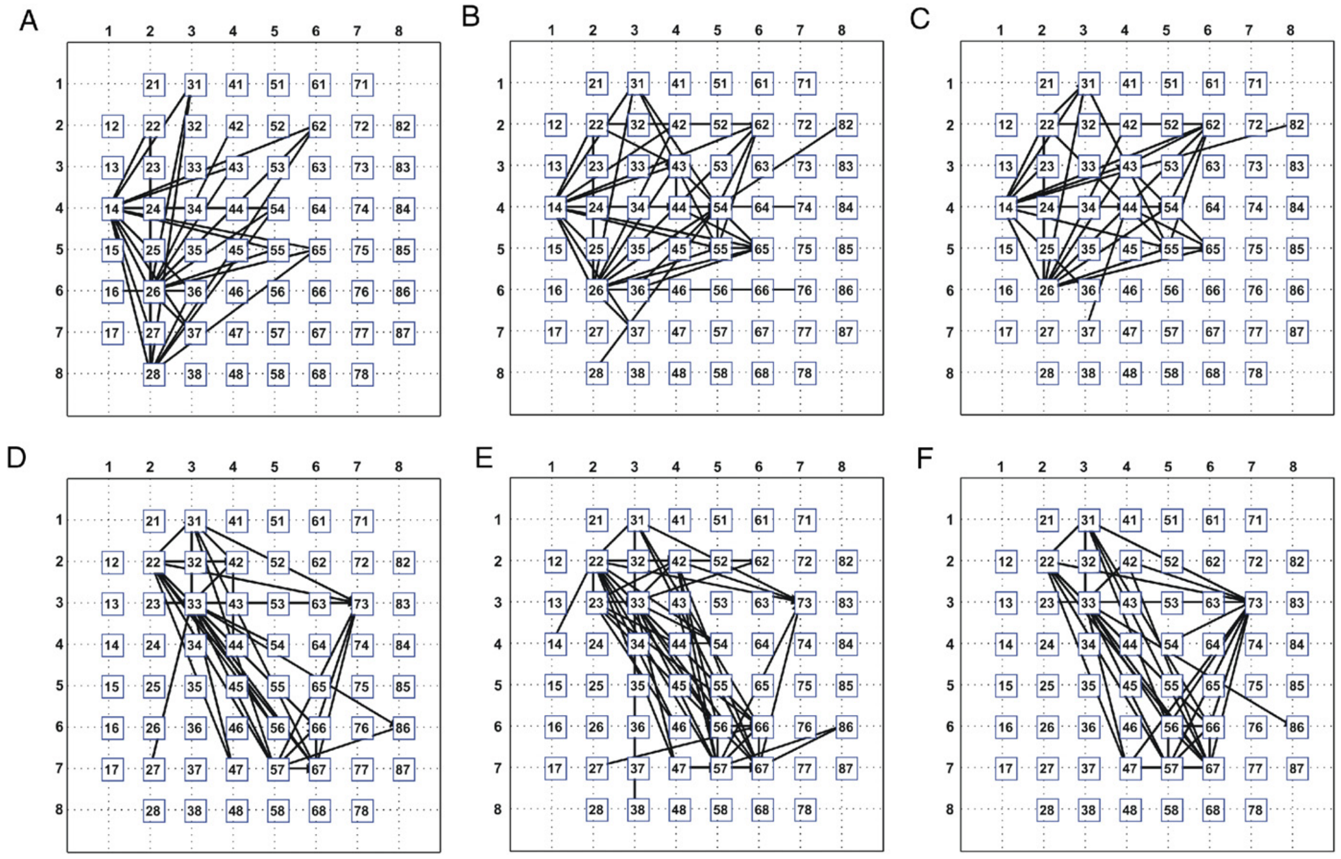


**Fig. 7.** Functional connectivity maps inferred by applying Transfer Entropy to the experiments of chemical manipulation: top, BIC 30 μM; bottom, APV 25 μM. The strongest sixty connections are identified and plotted to allow a better comparison among different phases.

all the chosen experiments all at once at the end of the analysis process.

## 4. Summary and conclusions

Due to the massive parallel recordings nowadays available, when analyzing multi-site neural signals, researchers must apply the same procedure (i.e., pre-processing, screening, statistical analysis, results display, etc.) to huge quantities of data. The experimental protocol is often repeated with small changes, like minor different pharmacological treatments or stimuli application vs spontaneous activity observations, or even with no-changes at all (e.g., only different date and length of observation). In such a scenario, nothing changes with respect to the analysis to be performed (e.g., spike detection, mean firing rate computation, burst analysis, etc.), because the interest of the researcher is to compare the same parameters in different experimental conditions. Nonetheless, many data have been collected and performing the analysis "by hand" to any subgroup would basically mean a waste of time. Things are even worse when several compounds are applied, imposing several control conditions monitoring, or when the activity of the cultures, either treated or not, must be followed throughout their lifetime. The last situation involves the collection of hundreds of hours of recording to be analyzed offline.

The above explanatory scenarios focus the attention on the main two needs suffered by scientists dealing with huge amounts of data. On the one hand, researchers aim at having a wide repertoire of analysis tools to explore new parameters to be used for subsequent statistical comparison; on the other hand, given that the same signal processing algorithms must be repeated several times, they are always searching for highly automated analysis software.

The software we presented here, addresses both the above needs. First of all it supplies the user with several analysis tools (both classical and innovative), able to explore the recorded activity from different points of view and making the detection of activity dissimilarities in apparently alike recordings easier. Secondly, it offers the possibility to batch process the data (i.e. multiple analysis), allowing the user to screen massive quantity of recordings from different experiments without continuously interacting with the software (e.g. in case of overnight processing).

Differently from other tools, either commercially available or presented in the literature (Cui et al., 2008; Egert, Knott et al., 2002; Meier et al., 2008; Wagenaar et al., 2005), SPYCODE was specifically designed to speed up the analysis phase and obtain, in a reasonable time, a series of maps and graphs to interpret the acquired multi-channel data from several experiments. The choice of Matlab as working environment was done to easily manage the folders containing the experiments' files and continuously upgrade the software with new functions and GUIs to be developed independently from the main interface. This strategy also envisages possible collaboration with partners from other institutions, working on multi-channel data, who can not only take advantage of SPYCODE for their data analysis but also develop their own tools and integrate them in SPYCODE with minimal efforts.

SPYCODE is available upon request. Please contact michela.chiappalone@iit.it.

## Acknowledgements

The authors are very grateful to Mariateresa (Brunella) Tedesco who performed the cell cultures and the experiments presented in this paper. The authors would like to thank all the people who, in the course of the years, have given a significant help in inspiring and developing the present version of the proposed software: Massimo Grattarola, Marco Bove, Giovanni Verreschi, Alessandro Vato, Laura Cozzi, Nicola Rosso and Antonio Novellino.

## References

Adrian, E. D. (1928). The basis of sensation: the action of sense organs. New York, NJ.

Arnold, F. J. L., Hofmann, F., Bengtson, C. P., Witmtmann, M., Vanhoutte, P., & Bading, H. (2005). Microelectrode array recordings of cultured hippocampal networks reveal a simple model for transcription and protein synthesis-dependent plasticity. *The Journal of Physiology*, *564*, 3–19.

Baker, R. E., Corner, M. A., & van Pelt, J. (2006). Spontaneous neuronal discharge patterns in developing organotypic mega-co-cultures of neonatal rat cerebral cortex. *Brain Research*, *1101*(1), 29–35.

Bakkum, D. J., Chao, Z. C., & Potter, S. M. (2008). Long-term activity-dependent plasticity of action potential propagation delay and amplitude in cortical networks. *PLoS One*, *3*(5), e2088.

Baljon, P. L., Chiappalone, M., & Martinoia, S. (2009). Interaction of electrically evoked responses in networks of dissociated cortical neurons. *Physical Review E*, *80*(3), 031906.

Baruchi, I., Volman, V., Raichman, N., Shein, M., & Ben-Jacob, E. (2008). The emergence and properties of mutual synchronization in in vitro coupled cortical networks. *European Journal of Neuroscience*, *28*, 1825–1835.

Beggs, J. M., & Plenz, D. (2003). Neuronal avalanches in neocortical circuits. *The Journal of Neuroscience*, *23*(35), 11167–11177.

Beggs, J. M., & Plenz, D. (2004). Neuronal avalanches are diverse and precise activity patterns that are stable for many hours in cortical slice cultures. *The Journal of Neuroscience*, *24*, 5216–5229.

Ben-Ari, Y. (2001). Developing networks play a similar melody. *Trends in Neurosciences*, *24*(6), 353–360.

Blanche, T. J., Spacek, M. A., Hetke, J. F., & Swindale, N. V. (2005). Polytrodes: high-density silicon electrode arrays for large-scale multiunit recording. *Journal of Neurophysiology*, *93*(5), 2987–3000.

Bologna, L. L., Nieus, T., Tedesco, M., Chiappalone, M., Benfenati, F., & Martinoia, S. (2010). Low-frequency stimulation enhances burst activity in cortical cultures during development. *Neuroscience*, *165*(3), 692–704.

Bonzano, L., Bove, M., & Martinoia, S. (2006). Effects of NMDA and non-NMDA receptors antagonists on the dynamic behavior of cultured cortical networks. *Neurocomputing*, *69*(16–18), 1897–1903.

Borst, A., & Theunissen, F. (1999a). Information theory and neural code [review]. *Nature*, *11*.

Borst, A., & Theunissen, F. E. (1999b). Information theory and neural coding. *Nature Neuroscience*, *2*(11), 947–957.

Bove, M., Grattarola, M., & Verreschi, G. (1997). In vitro 2D networks of neurons characterized by processing the signals recorded with a planar microtransducer array. *IEEE Transactions on Biomedical Engineering*, *44*(10), 964–977.

Buzsaki, G. (2004). Large-scale recording of neuronal ensembles. *Nature Neuroscience*, *7*(5), 446–451.

Chavez, M., Martinerie, J., & Le Van Quyen, M. (2003). Statistical assessment of nonlinear causality: application to epileptic EEG signals. *Journal of Neuroscience Methods*, *124*, 113–128.

Chiappalone, M., Bove, M., Vato, A., Tedesco, M., & Martinoia, S. (2006). Dissociated cortical networks show spontaneously correlated activity patterns during in vitro development. *Brain Research*, *1093*(1), 41–53.

Chiappalone, M., Massobrio, P., & Martinoia, S. (2008). Network plasticity in cortical assemblies. *European Journal of Neuroscience*, *28*, 221–237.

Chiappalone, M., Novellino, A., Vajda, I., Vato, A., Martinoia, S., & van Pelt, J. (2005). Burst detection algorithms for the analysis of spatio-temporal patterns in cortical networks of neurons. *Neurocomputing*, *65–66*, 653–662.

Chiappalone, M., Vato, A., Berdondini, L., Koudelka-Hep, M., & Martinoia, S. (2007). Network dynamics and synchronous activity in cultured cortical neurons. *International Journal of Neural Systems*, *17*(2), 87–103.

Cocater-Zilgien, J. H., & Delcomyn, F. (1992). Identification of bursts in spike trains. *Journal of Neuroscience Methods*, *41*, 19–30.

Corner, M. A. (2008). Spontaneous neuronal burst discharges as dependent and independent variables in the maturation of cerebral cortex tissue cultured in vitro: a review of activity-dependent studies in live 'model' systems for the development of intrinsically generated bioelectric slow-wave sleep patterns. *Brain Research Reviews*, *59*(1), 221–244.

Cozzi, L., D'Angelo, P., & Sanguineti, V. (2006). Encoding of time-varying stimuli in population of cultured neurons. *Biological Cybernetics*, *94*(5), 335–349.

Cui, J., Xu, L., Bressler, S. L., Ding, M., & Liang, H. (2008). BSMART: a MATLAB/C toolbox for analysis of multichannel neural time series. *Neural Networks*, *21*, 1094–1104.

Dayan, P., & Abbott, L. F. (2001). *Theoretical neuroscience: computational and mathematical modeling of neural systems*. Cambridge: MIT Press.

Egert, U., Heck, D., & Aertsen, A. (2002). Two-dimensional monitoring of spiking networks in acute brain slices. *Experimental Brain Research*, *142*, 268–274.

Egert, U., Knott, T., Schwarz, C., Nawrot, M., Brandt, A., Rotter, S., et al. (2002). MEA-Tools: an open source toolbox for the analysis of multi-electrode data with MATLAB. *Journal of Neuroscience Methods*, *117*, 33–42.

Eytan, D., & Marom, S. (2006). Dynamics and effective topology underlying synchronization in networks of cortical neurons. *The Journal of Neuroscience*, *26*(33), 8465–8476.

Garofalo, M., Nieus, T., Massobrio, P., & Martinoia, S. (2009). Evaluation of the performance of information theory-based methods and cross-correlation to estimate the functional connectivity in cortical networks. *PLoS One*, *4*(8), e6482.

Gourévitch, B., & Eggermont, J. J. (2007). Evaluating information transfer between auditory cortical neurons. *Journal of Neurophysiology*, *97*, 2533–2543.

Gramowski, A., Jügelt, K., Stüwe, S., Schulze, R., McGregor, G. P., Wartenberg-Demand, A., et al. (2006). Functional screening of traditional antidepressants with primary cortical neuronal networks grown on multielectrode neurochips. *European Journal of Neuroscience*, *24*(2), 455–465.

Gramowski, A., Jugelt, K., Weiss, D. G., & Gross, G. W. (2004). Substance identification by quantitative characterization of oscillatoryactivity in murine spinal cord networks on microelectrode arrays. *European Journal of Neuroscience*, *19*(10), 2815–2825.

Gross, G. W., Azzazy, H. M. E., Wu, M. C., & Rhodes, B. K. (1995). The use of neuronal networks on multielectrode arrays as biosensors. *Biosensors & Bioelectronics*, *10*, 553–567.

Gross, G. W., Rhoades, B. K., & Jordan, R. J. (1992). Neuronal networks for biochemical sensing. *Sensors and Actuators*, *6*, 1–8.

Gross, G. W., Rieske, E., Kreutzberg, G. W., & Meyer, A. (1977). A new fixed-array multi-microelectrode system designed for long-term monitoring of extracellular single unit neuronal activity in vitro. *Neuroscience Letters*, *6*, 101–105.

Hebb, D. O. (1949). *Organization of behavior*. New York: John Wiley & Sons.

Jimbo, Y., Tateno, Y., & Robinson, H. P. C. (1999). Simultaneous induction of pathway-specific potentiation and depression in networks of cortical neurons. *Biophysical Journal*, *76*(February), 670–678.

Kass, R. E., Ventura, V., & Brown, E. N. (2005). Statistical issues in the analysis of neuronal data. *Journal of Neurophysiology*, *94*, 8–25.

Knox, C. K. (1981). Detection of neuronal interactions using correlation analysis. *Trends in Neurosciences*, *4*, 222–225.

Lungarella, M., Pitti, A., & Kuniyoshi, Y. (2007). Information transfer at multiple scales. *Physical Review E*, *76*, 0561171–05611710.

Maccione, A., Gandolfo, M., Massobrio, P., Novellino, A., Martinoia, S., & Chiappalone, M. (2009). A novel algorithm for precise identification of spikes in extracellularly recorded neuronal signals. *Journal of Neuroscience Methods*, *177*(1), 241–249.

Maeda, E., Robinson, H. P. C., & Kawana, A. (1995). The mechanism of generation and propagation of synchronized bursting in developing networks of cortical neurons. *The Journal of Neuroscience*, *15*, 6834–6845.

Marom, S., & Shahaf, G. (2002). Development, learning and memory in large random networks of cortical neurons: lessons beyond anatomy. *Quarterly Reviews of Biophysics*, *35*(1), 63–87.

Meier, R., Egert, U., Aertsen, A., & Nawrot, M. P. (2008). FIND—a unified framework for neuronal data analysis. *Neural Networks*, *21*, 1085–1093.

Miller, E. K., & Wilson, M. A. (2008). All my circuits: using multiple electrodes to understand functioning neural networks. *Neuron*, *60*, 483–488.

Minerbi, A., Kahana, R., Goldfeld, L., Kaufman, M., Marom, S., & Ziv, N. E. (2009). Long-term relationship between synaptic tenacity, synaptic remodeling, and network activity. *PLoS Biology*, *7*(6).

Nicolelis, M. A. L. (2001). Advances in population coding. *Progress in Brain Research*, *130*, 49–62.

Nicolelis, M. A. L. (2003). Brain-machine interfaces to restore motor function and probe neural circuits. *Nature Reviews Neuroscience*, *4*, 417–422.

Pasquale, V., Martinoia, S., & Chiappalone, M. (2009). A self-adapting approach for the detection of bursts and network bursts in neuronal cultures. *Journal of Computational Neuroscience*.

Pasquale, V., Massobrio, P., Bologna, L. L., Chiappalone, M., & Martinoia, S. (2008). Self-organization and neuronal avalanches in networks of dissociated cortical neurons. *Neuroscience*, *153*(4), 1354–1369.

Perkel, D. H., Gerstein, G. L., & Moore, G. P. (1967). Neuronal spike train and stochastic point processes I. The single spike train. *Biophysical Journal*, *7*, 391–418.

Plenz, D., & Thiagarajan, T. C. (2007). The organizing principles of neuronal avalanches: cell assemblies in the cortex? *Trends in Neurosciences*, *30*(3).

Potter, S. M. (2001). Distributed processing in cultured neural networks. *Progress in Brain Research*, *130*, 49–62.

Quiroga, R. Q., & Panzeri, S. (2009). Extracting information from neuronal populations: information theory and decoding approaches. *Nature Reviews Neuroscience*, *10*, 173–185.

Raichman, N., & Ben-Jacob, E. (2008). Identifying repeating motifs in the activation of synchronized bursts in cultured neuronal networks. *Journal of Neuroscience Methods*, *170*(1), 96–110.

Raichman, N., Volman, V., & Ben-Jacob, E. (2006). Collective plasticity and individual stability in cultured neuronal networks. *Neurocomputing*, *69*, 1150–1154.

Rieke, F., Warland, D., de Ruyter van Steveninck, R., & Bialek, W. (1997). *Spikes: exploring the neural code*. Cambridge, MA: The MIT Press.

Salinas, E., & Sejnowski, T. J. (2001). Correlated neuronal activity and the flow of neural information. *Nature Reviews Neuroscience*, *2*(August), 539–550.

Schneidman, E., Berry, M. J., Segev, R., & Bialek, W. (2006). Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature*, *440*, 1007–1012.

Selinger, J. V., Kulagina, N. V., O'Shaughnessy, T. J., Ma, W., & Pancrazio, J. J. (2007). Methods for characterizing interspike intervals and identifying bursts in neuronal activity. *Journal of Neuroscience Methods*, *162*, 64–71.

Shannon, B. E. (1948). The mathematical theory of communication. *The Bell System Technical Journal*, *27*, 379–423.

Tam, D. C. (2002). An alternate burst analysis for detecting intra-burst firings based on inter-burst periods. *Neurocomputing*, *44–46*, 1155–1159.

Tam, D. C., & Gross, G. W. (1994). Extraction of dynamical changes in neuronal network circuitries using multiunit spike train analysis. In D. A. Stenger, & T. M. McKenna (Eds.), *Enabling technologies for cultured neural networks*. New York: Academic Press.

Tateno, T., Kawana, A., & Jimbo, Y. (2002). Analytical characterization of spontaneous firing in networks of developing rat cultured cortical neurons. *Physical Review Letters E*, *65*(5 Pt 1), 051924.

Van Pelt, J., Wolters, P. S., Corner, M. A., Rutten, W. L. C., & Ramakers, G. J. A. (2004). Long-term characterization of firing dynamics of spontaneous bursts in cultured neural networks. *IEEE Transactions on Biomedical Engineering*, *51*(11), 2051–2062.

Vato, A., Bonzano, L., Chiappalone, M., Cicero, S., Morabito, F., Novellino, A., et al. (2004). Spike manager: a new tool for spontaneous and evoked neuronal networks activity characterization. *Neurocomputing*, *58–60*, 1153–1161.

Wagenaar, D. A., DeMarse, T. B., & Potter, S. M. (2005). MeaBench: a toolset for multi-electrode data acquisition and on-line analysis. Paper presented in *The 2nd Intl. IEEE EMBS conference on neural engineering*.

Wagenaar, D. A., Pine, J., & Potter, S. M. (2006). An extremely rich repertoire of bursting patterns during the development of cortical cultures. *BMC Neuroscience*, *7*(11).

Xu, J., Liu, Z.-r., Liu, R., & Yang, Q.-f. (1997). Information transmission in human cerebral cortex. *Physica D*, *106*, 363–374.