NeuroCube

Realistic simulation of extracellular recordings using detailed neuron models

Overview:

A classic experimental approach in neuroscience is to study the spiking activity of individual neurons using microelectrodes inserted in the brain. The first step in the analysis of the data recorded with these electrodes is to detect the spikes of nearby neurons and classify their shapes into clusters corresponding to the firing of the different neurons, a procedure known as spike sorting. A very useful approach to evaluate the quality of a spike sorting algorithm is to use synthetic data, where the time and identity of each spike is known by construction. NeuroCube is a fast and detailed method to generate realistic simulations of extracellular recordings, where the simulations are obtained by superimposing the activity of neurons randomly placed in a cube of brain tissue recorded by a finite-size electrode.

Distribution and reference:

NeuroCube is free (and therefore without any warranty) for any non-commercial applications. For any commercial application, please contact the authors (lacm1@le.ac.uk & rqqg1@le.ac.uk). You can refer to this algorithm just by citing the paper where it is described:

A detailed and fast model of extracellular recordings.
Luis A. Camuñas-Mesa, Rodrigo Quian Quiroga
Neural Computation

This software was developed by Luis Alejandro Camuñas Mesa and Rodrigo Quian Quiroga.

Requirements:

NeuroCube runs under Windows, Linux and Mac. It requires Matlab 7.8 (R2009a) or higher.

Brief description of the code:

The code consists of five major steps:

i) Generation of the cube: first of all, it calculates the number of neurons included in the cube (given the size of the cube, the density of neurons and the proportion of active neurons), and then it places all these neurons randomly inside the cube by setting their coordinates. Depending on the distance to the electrode, each neuron is considered either as close-by or distant. Then, every close-by neuron is assigned a random neuron model (from the 20 possible set of parameters) and a firing rate (following the distribution set by the user). Using these firing rates and the duration of the simulation (set by the user), each close-by neuron is assigned a list of spike times, following a Poisson distribution. Every distant neuron is assigned a random spike shape from a database (*spike_shapes.mat*) and a firing rate (following the distribution set by the user). All this information is saved in a structure called *Neurons* that can be saved by the user in a mat file.

ii) Manual selection of single units: after obtaining the neuron distribution in the cube, the user can decide whether to keep the automatic result or to decide manually the number of single units. If the manual mode is selected, the user can activate up to 5 different neurons and specify the normalized distance and the firing rate for each one of them. In this manual mode, all the automatically generated neurons within a fixed distance to the electrode are deleted, and the manual neurons are introduced. The normalized distance is 0 for the closest possible distance to the electrode and 1 for the limit of the single unit volume.

iii) Electrode configuration: after that, the user can specify the electrode configuration, selecting in the GUI either a single electrode or a tetrode, the diameter and the separation between them. Any other electrode configuration can be specified by introducing the coordinates in the parameters of the file neurocube.m. Every time the electrode configuration is changed, the neuron distribution is updated according to the new distances to the electrode. If the number of channels is larger than one, the next steps will be replicated for each one of them.

iv) Calculation of spike shapes: in this step, it calculates the spike shapes associated to each neuron in the cube. For close-by neurons, that involves simulating the detailed model assigned to each neuron. For distant neurons, the spike shapes are just copied from the database, with their amplitudes scaled according to the distance to the electrode.

v) Generation of the final recording: in this last step, each distant neuron is assigned a list of spike times (using the firing rates calculated in the first step), following a Poisson distribution. After that, a train of spikes is generated for each neuron in the cube, and the addition of all the spike trains generates the final extracellular recording.

Data input and output:

There are two different ways to obtain the cube configuration: by setting the corresponding parameters in the GUI and clicking 'Generate Cube', or by loading a pre-saved cube. For the second option, an input **.mat* file can be used to load a certain cube configuration by clicking 'Load cube'. This file must include a structure named *Neurons* that includes the information needed to load that neuron distribution, with the following fields:

- Neurons.nneurons: total number of neurons in the cube.
- Neurons.coordinates: coordinates of all the neurons in the cube.
- Neurons.id: identification of the model that describes every neuron in the cube, where values <600 represent one of the spike shapes in the database for distant neurons, and values >600 represent the model assigned to that close-by neuron.
- Neurons.Rates: firing rates of all the neurons.
- Neurons.spiketimes: list of spike times for close-by neurons.

Any cube configuration can be saved as an output **.mat* file by clicking 'Save cube', including the *Neurons* structure.

After the extracellular recording is generated, an output **.mat* file can be saved by clicking 'Save'. That file will include:

- data: the continuous signal (or signals if more than one electrode is selected)
- Coordinates: coordinates of all the neurons in the cube.
- Coordinates_close: coordinates of only the close-by neurons.
- Neurons_id: identity of every neuron in the cube (for close-by neurons, it will indicate the models used to simulate them, while for distant neurons, it will represent the spike shape assigned to each one of them).
- Spiketimes: list of spike times (in milliseconds) for all the neurons in the cube.
- Close_neurons: matrix that indicates the neuron identification and spike time (in milliseconds) for every spike generated by close-by neurons.
- Close_Spikeshapes: shapes of the spikes generated by the close-by neurons.

Comments and updates:

If you have any comments please send them to me at lacm1@le.ac.uk. I really hope this algorithm will be useful for you. If it does, or if for some reasons it is not adequate for your data please let me know. I can't promise that I can introduce suggestions immediately, but I'll try to do it in a reasonable time. Also let me know by email if you want to keep updated on the release of any new versions, related papers, etc.