Quick guide to analysis

# These analysis are performed using MATLAB. If you have doubt about what you can do or how to do it please refer to [MathWorks documentation](ttps://www.mathworks.com/help/matlab/index.html) or ask the internet (someone had the same problems you have!). To understand this guide basic knowledge about MATLAB is advised (all questions are welcomed!).

# Clustering

From .continuous files to MATLAB arrays we can analyse.

##### Read the .continuous files to group of 4 electrodes before clustering

Create a folder within the current folder to save the new files.

Type the MATLAB function that generates those files:

1. For tetrode analysis:

read\_openephys('datadir', cd, 'resdir', [cd '\GR']);

1. For probe analysis:

read\_openephys\_probe('datadir', cd, 'resdir',[cd '\GR'], 'processor',101);

In this function, MATLAB will **read** files from the directory set by the option 'datadir', in the example *cd* corresponds to the current directory. The directory where MATLAB will **save** the files is set by 'resdir', in the example the folder named *GR* inside *cd*. The processor refers to the processor number set by the OpenEphys GUI to the saved channels. It’s automatically set as 101, but check it in the *settings.xml* within the folder. If you have rearranged the channel order using the **Channel Map** plugin make sure that this number corresponds to the processor you analyse.

For probe analysis (B) the recording sites are saved as groups of 4 channels, named GR + number of group; and groups of 2 channels for the in-between groups, named Pair + number of pair. To match your probe, you can open the function

open read\_openephys\_probe

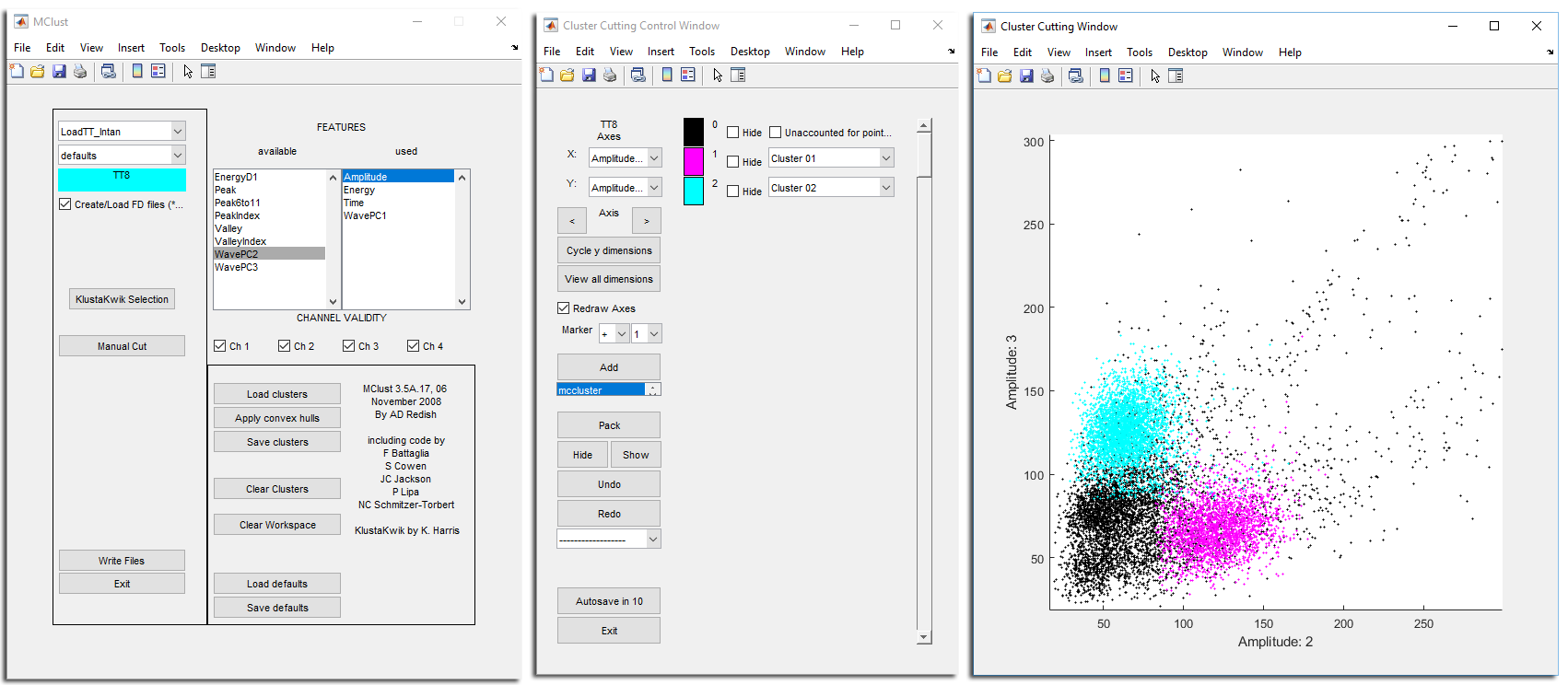
and change the number of groups of electrodes (TTspec) and number of channels (CHspec) adding to the command function: 'TTspec', 1:8 (where 8 is the number of groups) and 'CHspec', 32 (where 32 is the number of channels).

In both cases, the spike discrimination is set by a voltage threshold (Th). This voltage can be changed opening the function and, in line 39, changing the value (30).

Th = 30; % Threshold for the spike detection

Once the function finishes, the previously created folder should contain the extracted waveforms and timestamps of the putative neurons in .mat files (named GR + number of group) and the calculated features that will be used by the MClust in .fd files.

##### Using MClust to separate the putative neurons



Loading engine

Loaded data

Click to load data

Used features of the neurons

Number of electrodes to load

Clear to load new data

Save the timestamps of the neurons in .mat files

To cluster the neurons

Load clusters

Save clusters

To open the MClust just type the command in the MATLAB Command window.

MClust

A new window will pop out (see Fig 1). Quick steps:

1. Make sure the loading engine selected is LoadTT\_intan.
2. Make sure that the channel validity (number of electrodes to load) is correct. You can activate/ deactivate channels.
3. The Features panels show the used and available features to use while clustering. Click on the features to add/ remove.
4. Click Create/ Load FD files and select the group you want to analyse.
5. Once loaded (the box turns blue and the name of the data appears) you can: A) Load previous clusters (Load clusters); or B) Make new clusters (Manual cut). If there are errors during the load the box will be red and the name of the data will not appear, check Channel Validity, Clear Workspace or Exit and Open again the MClust.

Figure 1. MClust main window.

1. To make or modify clusters click on Manual cut.
2. Once the clustering is finished save them clicking on Save clusters.
3. To save the .mat files containing the timestamps of the neuron click on Write Files. This saves all clusters in the memory to .mat in the order you clustered them. The name of the saved files will be the number of the group + \_ + cluster number (i.e. GR2\_1).
4. Before loading the next group, clear the memory by clicking Clear Workspace.

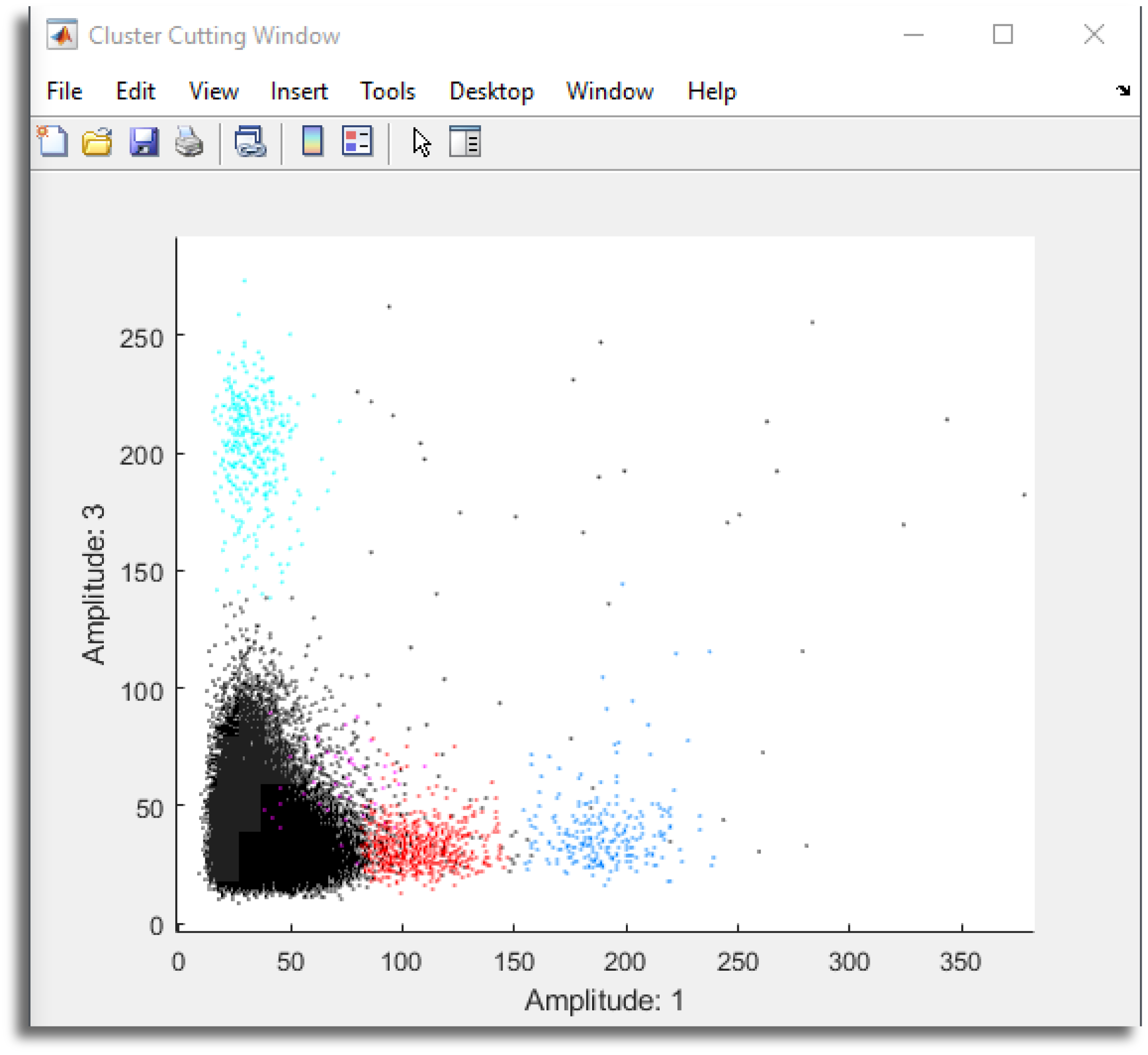
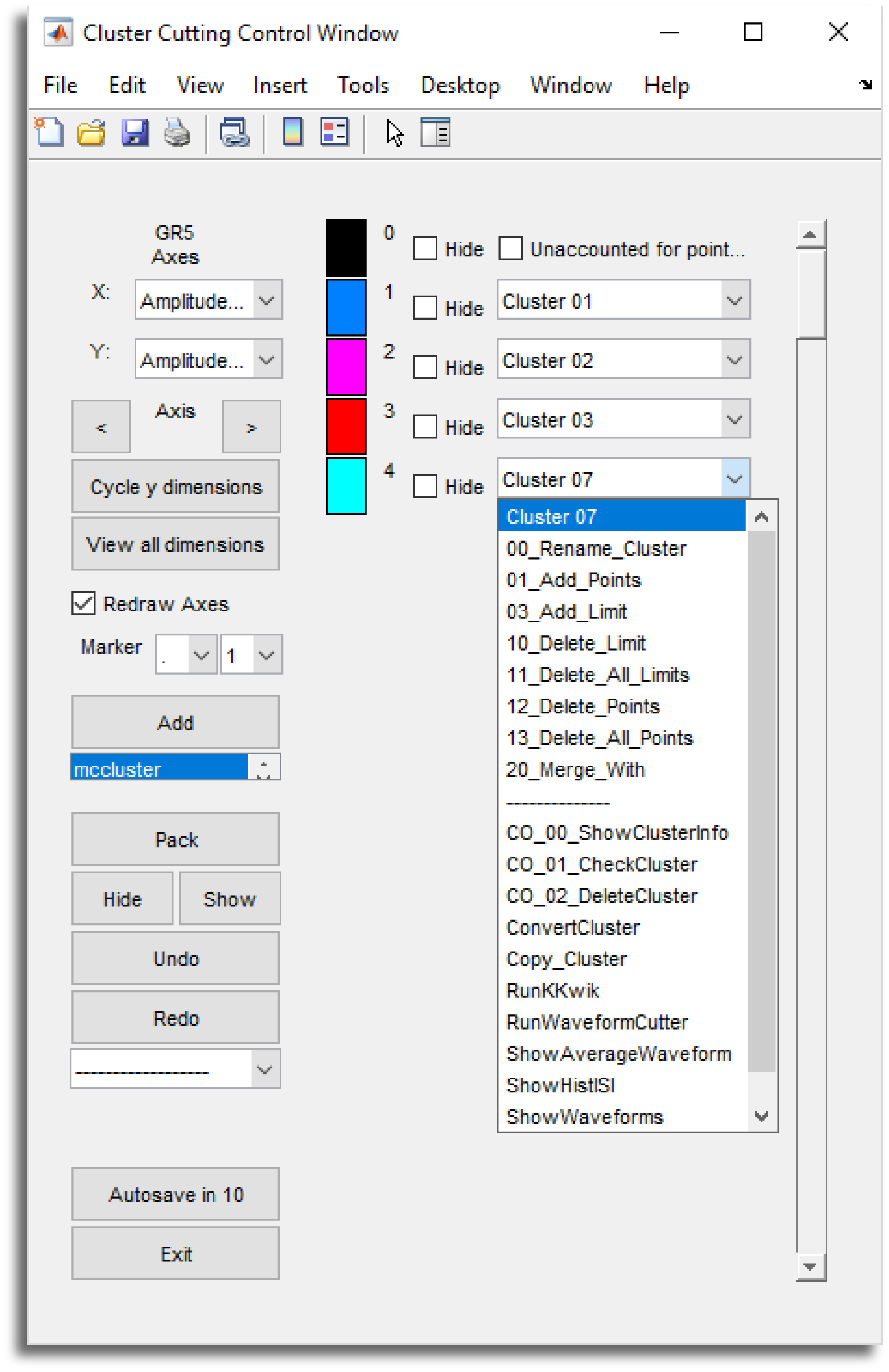
###### Clustering

To cluster the neurons select Manual cut. The Cluster Cutting Control Window will pop up (Fig. 2). The order of the steps shown below does not reflect the actual workflow, you can use them repeatedly and in any order.

Quick steps:

1. Plot the features by clicking Redraw Axes, while the box is clicked all changes you make in the plot (Cluster Cutting Window) will automatically appear.
2. To add a new cluster press Add.
3. To add dots to the cluster, in the drop-down menu on the cluster name, press 01\_Add\_Points. Then move the cursor to the Cluster Cutting Window and draw a circle (clicking on the plot) around the cloud.
4. Change the plotted features to better define the cloud. You can, in the drop-down menu on the cluster name, 01\_Add\_Points, 03\_Add\_Limit (create a circle -by clicking- that excludes points outside), 10\_Delete\_Limit, 12\_Delete\_Points.

Figure 2. MClust cutting window. Left, Main cutting window; Right, Features plot.



Y / X features displayed

Press to change the features

Click to plot

Add cluster

Undo/ Redo changes

To convert clusters

Clusters, click/ unclick to hide them in the plot

Possible actions to do to a cluster

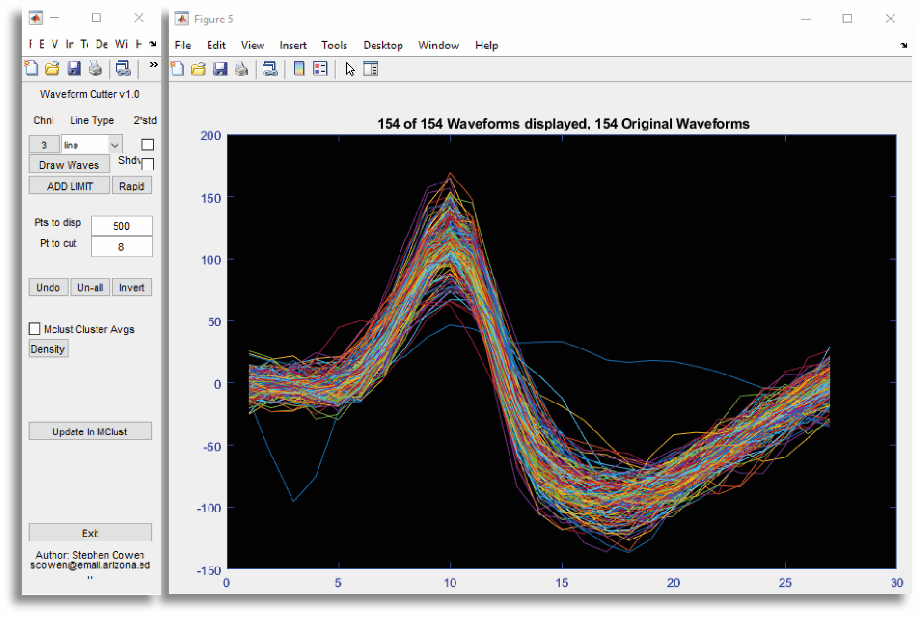
X feature

Y feature

Clouds of dots are putative neurons

# Cluster

1. Define the cloud limiting by the waveform with RunWaveformCutter (Fig. 3), in the drop-down menu on the cluster name. Select the channel were you will limit the waveform, if a neuron appears in more than one channel check them all. Different views will give you better idea of the waveform, also you can Draw Waves to redraw the waveforms showed. To limit the waveform click on Add limit (one) or Rapid (more than one), then on the plot click the TOP and BOTTOM limits for the specific part of the waveform (yellow dashed line). I recommend to limit the peak, valley and the ascending or descending parts. If it requires too many limits, maybe it’s not properly isolated. To save the changes press Update in MClust, now you can Exit.



Channel number

Select plot view

Restrict waveform once

Save

Restrict waveform, repeatedly

Click to limit waveform shape

Figure 3. MClust waveform cutting window. Left, Control window; Right, Cutting window.

1. Check the interspike interval using ShowHistISI, in the drop-down menu on the cluster name. The number of violations of the ISI, try to avoid spikes under the red dashed line.
2. For an automatic clustering use RunKKwik, in the drop-down menu on the cluster name. Add a new cluster, add points (usually they correspond to more than one cloud), press RunKKwik. In the new window (Fig. 4), select the min and max number of clusters (use the number of clouds you see as reference and add 2 more just in case); then add the features to use of the analysis and GO. This process takes some time. As an idea, you can use it for difficult to cluster channels and only add the features of that channel. After the automatic cluster finishes check the isolated clouds, ISI and Waveforms. Automatic is easier but not perfect. To check some features you first need to convert the clusters: click on the drop-down menu under Redo, select ConverAllClusters and, in the new window, select @mccluster. IMPORTANT, always check that the clusters seem logical (and what you would cluster as one neuron is in one cluster).

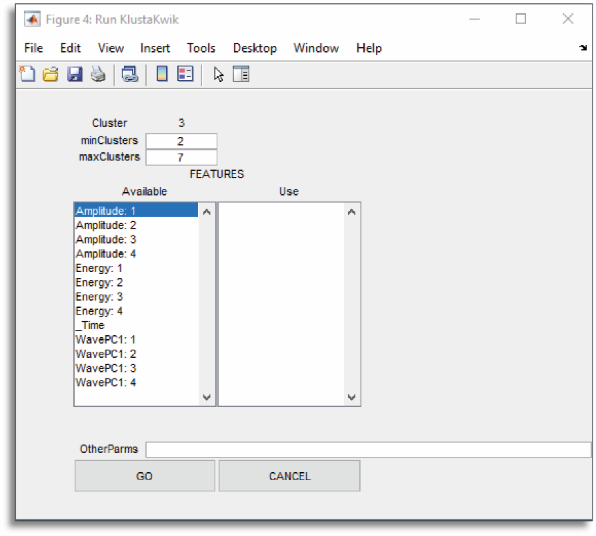


Figure 4. MClust KKwik window.

Min/ Max num of clusters

Used features

1. Check the quality of the cluster using CO\_01\_CheckCluster, in the drop-down menu on the cluster name. This provides:

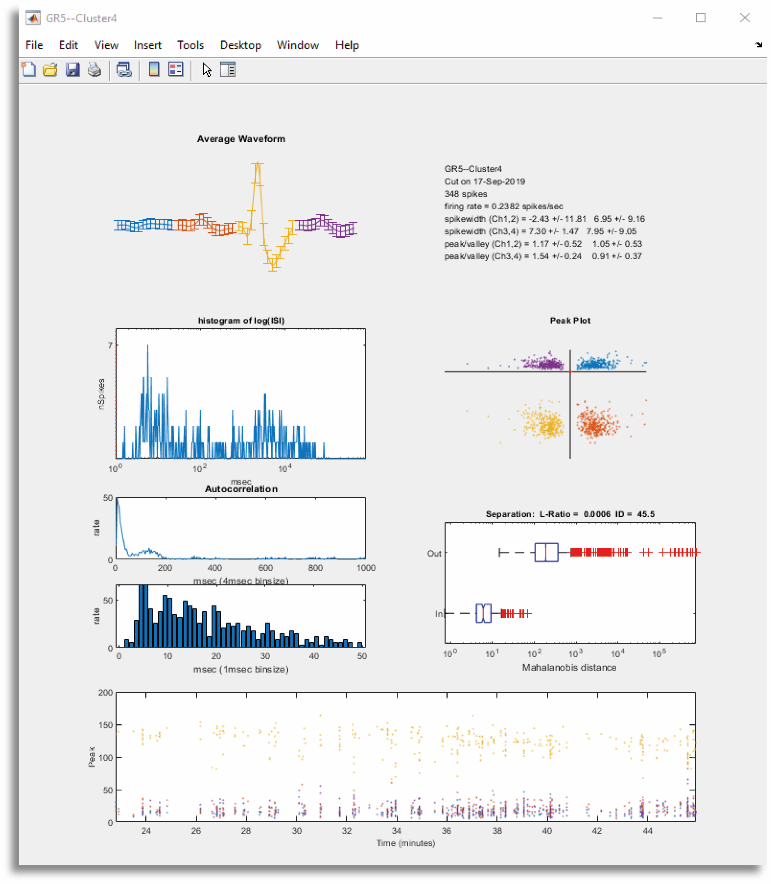


Figure 5. MClust Check cluster window.

Waveforms

Peak plot

ISI histogram

The math

Timestamps

Autocorr

* 1. a plot of the mean + std of the waveforms; the std should not be too wide, as this indicates it might not be properly isolated.
  2. ISI histogram, as previously.
  3. autocorrelogram, if there are no violations of the ISI these should show a blank activity after 0.
  4. peak plot, expected to be an elliptic cloud of dots, if not then the cluster is not well limited.
  5. the Math, gives actual mathematical values about the goodness of the cluster; the **L-ratio** measures the probability that the clustered cloud is separated from the rest of the detected spikes (values under 0.15 are acceptable and under 0.05 better); the **Mahalanobis distance (ID)** measures the distance from the centroid of the cluster to the rest of the spikes (usually values over 20). Remember that biology likes to mess with the math so take these values as reference, but don’t discard things that look like neurons and behave like neurons just because the numbers are not the best.
  6. Firing of the neuron along time, neurons tend to fire with some regularity and don’t usually appear and disappear. The Check Cluster is set to use 4 channels by default, if you wish to check only 2 channels open the function

open CheckCluster.m

and change the channel validity (ChV) in line 52.

1. After finishing the clustering exit the Cluster Cutting Window pressing exit. In the main window you can save the clusters (see Fig. 1).

# Load data to MATLAB

##### Read a .continuous to MATLAB

To read the information from a .continuous file to a MATLAB variable so you can work with it. Type

[data, ts, ~] = load\_open\_ephys\_data('101\_ADC1.continuous');

this function reads the file and generates the data variable and the timestamps variable. With these two you can do whole data analysis or select specific parts to analyse.

##### Read an Intan .rhd to MATLAB

To read the information from an .rhd file recorded with the Intan Acquisition Board to a MATLAB variable so you can work with it. Includes 2 options:

OPTION A: tell the file and path (if you use cd it will use the current directory) of the file to analyse. Type

read\_Intan\_RHD2000\_file('FileA.rhd', 'L:\Cecilia\rhd\_files')

or

OPTION B: pop up window to select the file. Type

read\_Intan\_RHD2000\_file

this function is a modification of the Intan provided function. It generates a data variable and time variable for the recorded channels and/ or ADCs.

##### Load TTL data

To generate a .mat file with the TTLs. Includes 3 cases of getting the TTLs.

OPTION A: using the signal recorded with OpenEphys (default).

TTL = load\_ttl('101\_ADC1.continuous');

OPTION B: using the signal recorded with Intan, previously saved in a .mat file. IMPORTANT, type 'origin', 'IN' in the varargin.

TTL = load\_ttl('TTL.mat','origin','IN','n',60,'mPP',0.5,'mPD', 0.1, 's');

OPTION C: using the data in the Workspace (e.g. the audio extracted from a video). IMPORTANT, type 'origin', 'WS' in the varargin.

TTL = load\_ttl('origin','WS','n',60);

this will create a vector variable (TTL) with all the times that cross a certain threshold. Several additional features are available: number of TTLs, change the threshold, the distance between peaks, auto-save. To know more read the function Help.

# Using MATLAB to analyse the neurons

Now you have all the things you need to start the analysis itself. The following list provides a general idea of the available functions and how to use them. Remember, if you have new code you can add it to this list and if you have more things you want to analyse you can challenge yourself (or me) and code it.

##### Analyse the waveforms

Choose the neuron you want to analyse (e.g. 'GR5\_1', that is located in group 'GR5'). Type

wf = waveform\_analysis('GR5.mat', 'GR5\_1.mat', 'features', 'plot', 'mean', 's', 'jpg');

or

waveform\_analysis('GR5.mat','GR5\_1.mat');

this function calculates the features of the mean waveform and/ or plots the waveforms of the channel where the spike is detected. Several additional features are available: plot only mean waveform or all of them, calculate the features, autosave the plot. To know more read the function Help.

##### Analyse the interspike interval

Choose the neuron you want to analyse (e.g. 'GR5\_1'). Type

[ISIh, ISIbins] = ISI\_hist('GR5\_1.mat','bins',500,'s','jpg');

or

ISI\_hist('GR5\_1.mat')

this function calculates the interspike interval of the timestamps of the neuron. Several additional features are available: modify the bins, autosave the plot. To know more read the function Help.

##### Analyse the auto-correlogram of a neuron

Choose the neuron you want to analyse (e.g. 'GR5\_1'). Type

[HISTvals, X] = ACorrelogram('GR5\_1.mat', 'bins', 1, 'width', 250, 's', 'jpg');

or

ACorrelogram ('GR5\_1.mat','s', 'jpg');

this function calculates the auto-correlogram, which compares the firing times of one neuron versus the neuron itself. Is like the ISI histogram except that each spike is compared not just with adjacent spikes but with all other spikes falling within the temporal range of the histogram, thus is useful for revealing multiple periodicities within a spike train. Several additional features are available: modify the bins or width, autosave the plot. To know more read the function Help.

##### Analyse the cross correlogram between two neurons

Choose the neurons you want to analyse (e.g. 'GR5\_1', 'GR8\_1'). Type

[Y, X] = XCorrelogram('GR5\_1.mat', 'GR8\_1.mat', 'bins', 1, 'width', 250, 's', 'jpg');

or

XCorrelogram('GR5\_1.mat', 'GR8\_1.mat', 's', 'jpg');

this function calculates the cross correlogram, which compares the firing times of one neuron versus another to check whether their firing probability has a fixed delay and, thus, they are related. Several additional features are available: modify the bins or width, autosave the plot. To know more read the function Help.

##### Analyse the peri-stimulus time histogram

Load the timestamps of the desired neuron and the TTL of the event you want to analyse. Then type

[psth, ts] = ttl\_psth (timestamps, ttl, 1000);

or

[psth, ts, psth1st, ts1st] = ttl\_psth (timestamps, ttl, 1000, 'pre', 0.5, 'post', 0.5, 'chart', 2);

this function generates a peri-stimulus time histogram and returns the histogram bars (psth) and the spike timestamps (ts) relative to trigger times, it can also return the values up to the 1st spike after the TTL (psth1st, ts1st). IMPORTANT: all timestamp inputs (timestamps, ttl) must be in the same units, ideally seconds. There are several varargin you can modify to match your recording. To know more read the function Help. This function relies in other two psth\_hist and psth\_raster to generate the plots.

##### Analyse the psth of all the neurons in a folder

Go to the folder you want to analyse and check the number of groups to analyse. Then type

psth = allneurons\_psth([1:8], ttl);

or

psth = allneurons\_psth(2, ttl, 'pre', 0.5, 'post', 0.5, 'fr', 30000);

this function calculates the psth of all the neurons within the selected groups and returns a structure with the spikes around the event (all and up to 1st spike), and the mean values for the firing times of the neuron. You can choose to analyse a specific group (bottom) or a range of groups (top). To know more read the function Help.

##### Analyse a neuron

Choose the neuron you want to analyse (e.g. 'GR5\_1') and type

A) neuron\_analysis('GR5\_1.mat');

or

B) neuron\_analysis('GR5\_1.mat', 'wf', 'plot', 'mean', 'ISI');

this function calculates several data from a neuron: features of the waveforms, plots the waveforms, ISI histogram, cross correlation and autocorrelation, autosave the plot. You can choose which things to calculate (B), while the default mode calculates everything (A). To know more read the function Help.

##### Analyse all neurons in a folder

Go to the folder you want to analyse and check the number of groups to analyse. Type

A) ALLneurons\_analysis(8);

or

B) ALLneurons\_analysis(8, 'wf', 'plot', 'mean', 'ISI');

where 8 is the number of groups. This function calculates several data from each neuron (located in groups 1 to 8): features of the waveforms, plots the waveforms, ISI histogram, cross correlation and autocorrelation, autosave the plot. You can choose which things to calculate, the default mode calculates everything (A).

Actually this works only for the analysis of the groups. If you want to analyse the neurons in the pairs, please use the function neuron\_analysis. To know more read the function Help.

# Using MATLAB to analyse the LFP

Starting with the raw data.

##### Fourier analysis

Choose the signal you want to analyse (e.g. data, read from the desired channel). Type

[FFT, fq] = fourier\_analysis(data,[0.5 300]);

or

fourier\_analysis(data,[0.5 300],'mother', 'Multi-taper');

this function calculates the spectrum density (e. g. frequency) of the time-domain of the signal and provides a plot and the calculated variables (if needed). Several additional features are available: sampling rate, down-sampling factor, mother wavelet function. To know more read the function Help.

##### Wavelet analysis

Choose the signal you want to analyse, in this case you need both the data and time variables (e.g. data and ts, read from the desired channel). Type

[wdata, wfreq] = wavelet\_analysis(data, time,[100 300],[0.5 300]);

or

wavelet\_analysis(data, time,[100 300],[0.5 300], 'fr', 20000, 'dj', 0.05, 'n', 5, 'mother', 'Morlet', 'tInt', 100);

this function calculates the continuous wavelet transform analysis, which is similar to the FFT, but shows the temporal evolution of the dominant frequency. Because is very memory consuming you need to restrict the time to analyse (e.g. [100 300]) and the frequency range (e.g. [0.5 300]). Several additional features are available: sampling rate, resolution, downsampling factor, mother wavelet function and segmenting intervals. To know more read the function Help.

# Other useful MATLAB scripts

Starting with the raw data.

##### Convert .continuous files to .dat files

Specially thought to read .dat files to Spike2. Please remember that the data import type should be signed 16 bits.

OEtoDat;

or

OEtoDat('processor',101);

From the current directory checks for the files named 101\_CH\* and 1) generates a common average of all the channels; 2) reads each channel, subtracts the common average and saves as CH\*.dat. If the processor number of the recorded OE data is not the 101 you can select another (as in the bottom example).