Primary neuronal culture exhibits spontaneous network patterns monitored by multi-electrodes array

Internship report from Raphaël Tinarrage

August 2014

" I don't think the brain came in the Darwinian manner. In fact, it is disprovable. Simple mechanism can't yield the brain. I think the basic elements of the universe are simple. Life force is a primitive element of the universe and it obeys certain laws of action. These laws are not simple, and they are not mechanical."

Kurt Gödel

Acknowledgment

In the context of UPSUD's MFA Magistère, I did an internship, during the month of August 2014. I worked under the direction of Jérémie Sibille, hosted in Nathalie Rouach's laboratory at the Collège de France, in Paris.

I'd like to adress all my thanks to Jérémie Sibille, for introducing me his fabulous field of research, and for all the time he took to teach me.

Introduction

Primary neuronal cultures exhibit spontaneous rhythmic networks activity after two weeks *in vitro*. Recorded by Multi-Electrodes Array (MEA), this emergent rhythmic activity depends on the connections of precursor neuronal cells, and undergo from both excitatory and inhibitory neurons. We here aim to better decipher the nature, properties and characteristics of these stereotypic activities.

The aim of this internship was to built a Matlab code to extract, analyse and quantify this typical spontaneous neuronal networks rythmic activity. I had the opportunity of discovering the exciting world of experimental neurobiological research, and even trying dissection.

Contents

Chapter 1

Biological experimental background

1.1 Monitoring neuronal physiology

In the last half century, neurons has been studied mainly by monitoring their membrane potentials (and all related pathways). **Electrophysiology** was firstly developp in the 50's by intra axonal recoding in the giant squid axons. This technic allowed to decipher the exact timing of an action potential. Today, single neuron activity and population of neurons can be recorded by different means, including magnetic resonnance Imaging, Positron Tomography, Electroencephalogram and electrodes inserted in the brain for living animals (*in vivo*). Moreover **electrophysiology** is extensively performed on acute slices (*ex vivo*), by several means such as dendritic patch, loose patch, whole cell or meaned field recordings. Those technics can also be used for *in vitro* studies.

We cultivated neurons above Multi-Electrodes Arrays (MEA), which record the mean field variations (see figure 1.1). This set-up allows us to observe both the the population activity and the single neuronal firing embedded in, basis of all known cognitive processes.

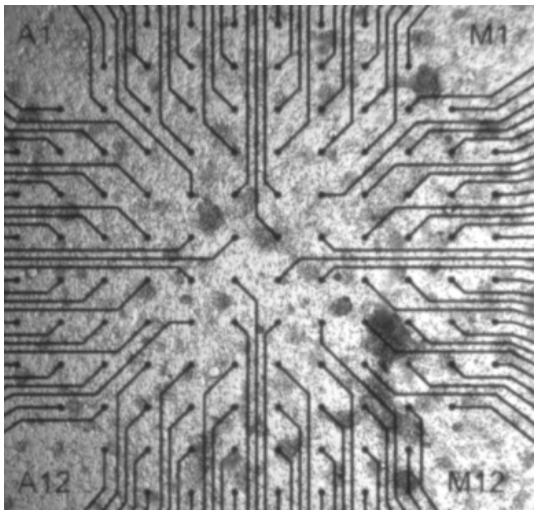

Figure 1.1: Photography of a primary neuronal culture on MEA at DIV 19 *(days in vitro)*. Note the below confluence concentration and the presence of few point of higher neuronal concentration (blob)

1.2 Primary neuronal culture preparation (material and methods)

More precisely, primary neuronal cultures were prepared as previsouly published (Gullo et al. 2009, Berdondini et al. 2009) from hippocampus of E18 mice embryos (OF1 background). Once the hippocampi are extracted from embryos, they are cut with a blade and incubated in NeuroBasal medium solution* with 0,5% Trypsin for 25 minutes (Neurobasal* is a commercial Neurobasal solution complemented with B27, Glutamatax, Penicyline/Strepptomycin and Fongizone). Neurons are then bathed in Neurobasal* complemented with SVG serum for 5 min. The following dissociation is reqest for a good preservation of neuronal extracellular organells by gently pressuring Neurobasal* complemented with DNAses (50μ/mL) in order to partially dossicated the melted hippocampal tissue. After few pressurisation the supernatant should contains a higher enough concentration of singled neurons in solution (up to 750 000 cell/mL) that is separated from the remaining tissue. This step is repeated as much as necessary. At last, 8

to 12mL of neurons suspended in Neurobasal* is obtained. This suspended solution is plated in precoated MEA. The coating of the MEA consists in 1mg/mL Poly-D-Lysin solution over night, dried, and then plated for 2h at 37°C with 20 μg/mL Laminin solution). The totality of the medium is changed two hours after plating and then one third of the medium is change every three to four days. All chemical are purchased in Gibco corp, MEA and MEA amplifier are puchased from Multichannels system (MEA 2100 amplifier for 120 channels) and recorded in a dry CO2 incubator.

After a week, the neuronal precusor cells have already started their differentiation, in excitatory or inhibitory neurons (see figure 1.2). Two weeks after plating, the neuronal population starts to exhibit spontaneous neuronal network activities, repeating patterns in rhythm.

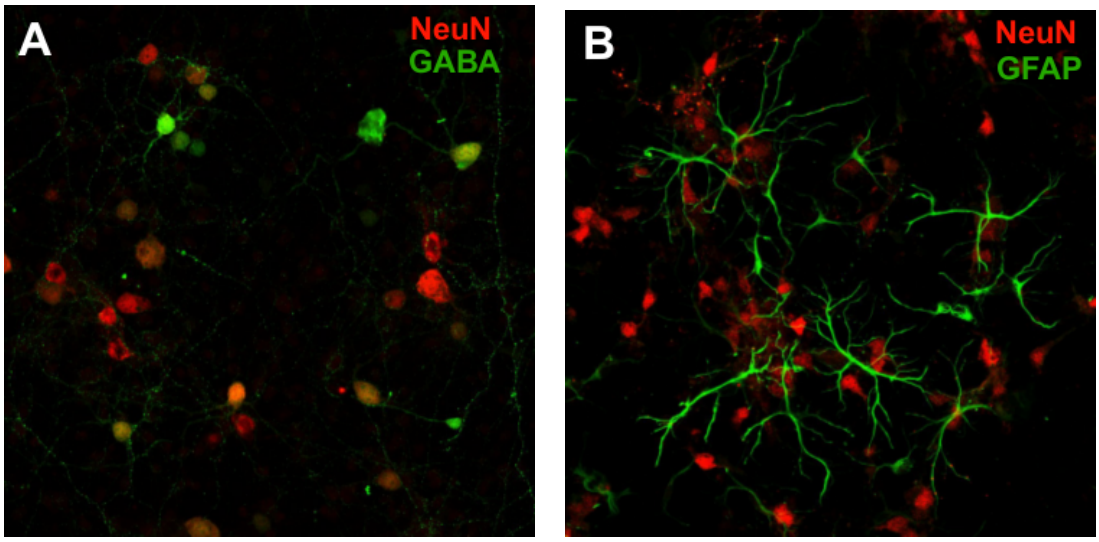

Figure 1.2: Immunofluorescence of obtained culture: postivie neurons to both GABAR and NeuN are present. These culture were made from sister culture, plated on glass cover slips, and observed in a confocal microscope. In other words, the cells have been equipped with fluorescent markers depending on their proteic expression (characteristic of their cell type). **A** : Picture of the culture showing both excitatory and inhibitory neurons (antibodies NeuN and GABAR). **B** : Picture of the culture showing the presence of glial cells next to the neurons.

1.3 Signals obtained from neuronal culture on Multi Electrodes Array

A neuron, when getting activated, show depolarization of its membrane potential. If this depolarization reaches its threshold of firing, then it fires an action potential. We monitor these activities in our culture by recording the meaned field on each electrodes of the MEA. The MEA we used comes from the German company **multichannel systems**. Each MEA counts 120 channels, and our recordings were made at a sampling frequency of 25000 Hz (see figure 1.3).

The events we extracted from the mean field recording on each channels were **action potentials** (see figure 1.4).

The second type of studied activity were **bursts**, which are the trace of a strong synchronous networks activation, characterized by many synchronous action potentials recorded on a majority of the channels. Therfore we quantified these phenomenon by sliding a time window summing all the events (as illustrated in 1.5).

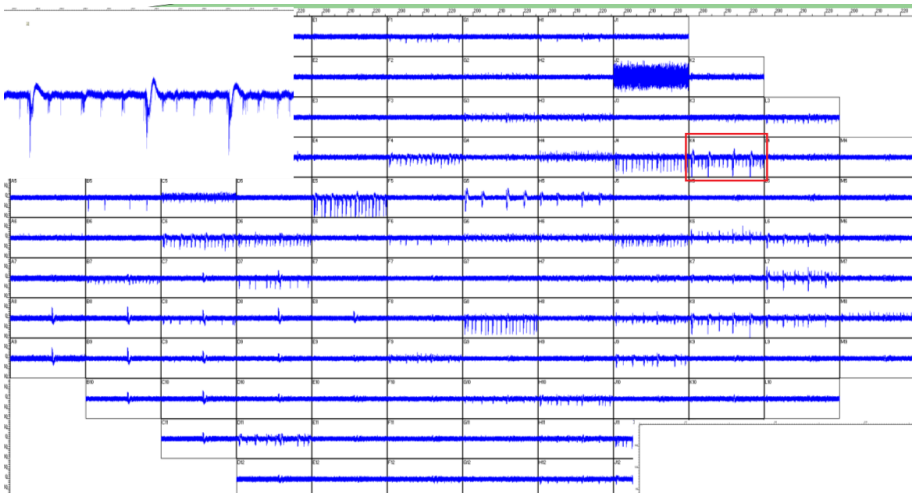

Figure 1.3: Illustration of a software captation: overview of mean field recordings of the 120 sites of the MEA for a 30 seconds period. Note the presence of bursts of different sizes and involving different sets of channels. **Inset above left** : recording from channel K4 (highlighted in red) illustrating the different type of burstig patterns occuring on a single channel.

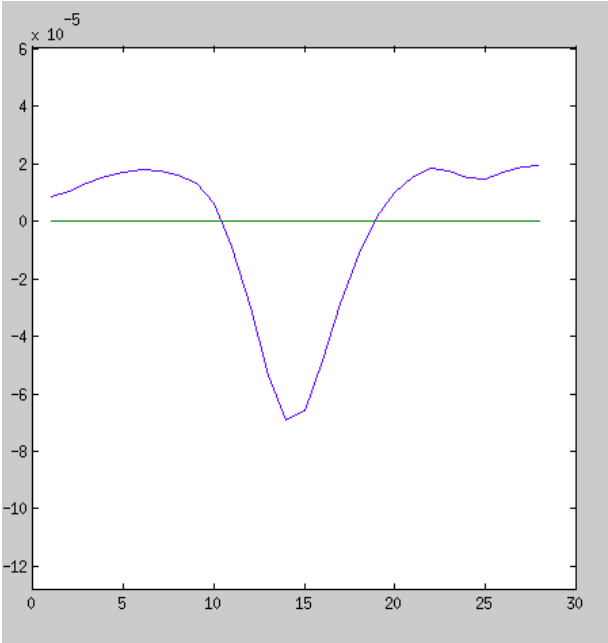

Figure 1.4: Local meaned field recording obtained on a single recordings site during an action potential. Note the particular shape obtained during this recordings, which corresponds to the different steps of the membrane potential changes during an action potential.

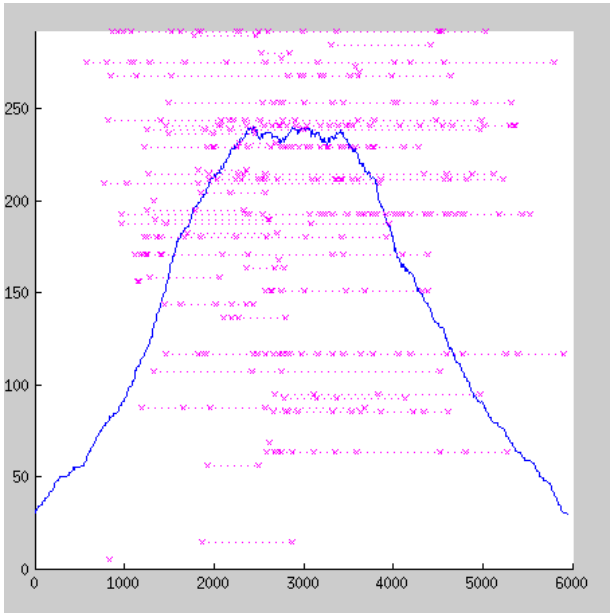

Figure 1.5: Shape of a burst. The valence (blue) represents the global behavior of the culture during the burst, and a raster plot (pink) illustrates the firing in each channels.

Chapter 2

Data analysis : extraction and quantifications

The main objective of this internship was to elaborate a Matlab code to extract, visualize and analyse the MEA recordings. We wanted this code to be a *naive* toolbox to display and partially process the data.

2.1 Overview of the code's main functions

During this internship, I realized a GUI (Matlab graphical user interface), which permits to treat the raw data in different ways :

- The function PROCESS, which converts a MCD file into a workable Matlab file
- The function VISUAL, which displays choosen values and graphs from the processed files
- The function CAT, which permits to concatenate several recordings

This was done in order to ease the handling and treatment of several experiments coming from the biological lab I was working in. The task of this GUI is first to extract the raw signal from the .MCD database. Concatenation of several recordings can be done when necessary. Once converted into matlab workable files, the goal is first to extract action potentials features in order to describe the burst occurence. Therfore the visual function contains two different options builted for either Action potential visualization or Burst visualization. In addition, PROCESS also contains the function ROUTINE, which permits to treat all the MCD files contained in a folder and its subfolders, and put away the output files in a new folder which reproduces the original tree view.

The function PROCESS returns a structure, gathering all the features we extracted from the recordings. Here is the list of the output structure content:

- Recording name
- Count of channels
- Action potentials
  - Number of the action potentials
  - Start time
  - Length
  - Values of the recording
  - Min
  - Left peak
  - Rigth peak
  - Halway length
  - Ratio of potentials before and after the event
- Bursts
  - Count of involved action potentials

- Start time
- Length
- Peak of the valence
- Values of the valence
- Rasterplot

2.2 Extraction of Action Potentials

Thanks to the library *Neuroshare*, given a MCD file, Matlab is able to access to the recorded signal of each channel. Therfore each channels will be treated independently. The main point of the processing is to extract the action potentials, and to avoid the noise.

2.2.1 Filtering and thresholding the raw recordings

The first step was to rid the signal of the low frequencies, which correspond to second-long depolarization of the local field's baseline. To do so, we use a high-pass *Chebyshev filter* (broadly used in electrophysiology).

Here is the expression of this fitler :

$$H(i\omega) = \frac{0,8751 - 3,4916i\omega + 5,233(i\omega)^2 - 3,4916(i\omega)^3 + 0,8751(i\omega)^4}{1 - 3,7238i\omega + 5,2175(i\omega)^2 - 3,2594(i\omega)^3 + 0,7658(i\omega)^4}$$

The raw signal and spectrum are illustrated in the figures 2.2.1 and the result of the filter 2.2.1.

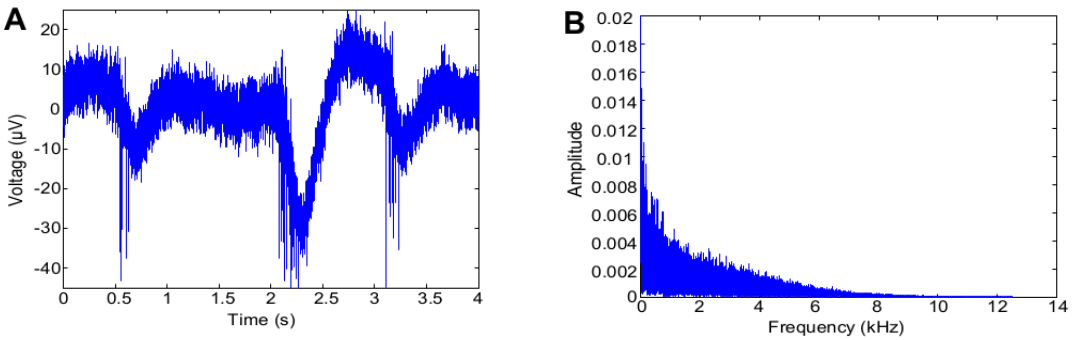

Figure 2.1: Values and spectrum of the raw signal: A: Raw signal obtained on a single recording site. Note the combination of slow wave oscillations with rapide events (Action potentials). B: Corresponding spectrum obtained by Fast Fourrier Transform (FFT).l

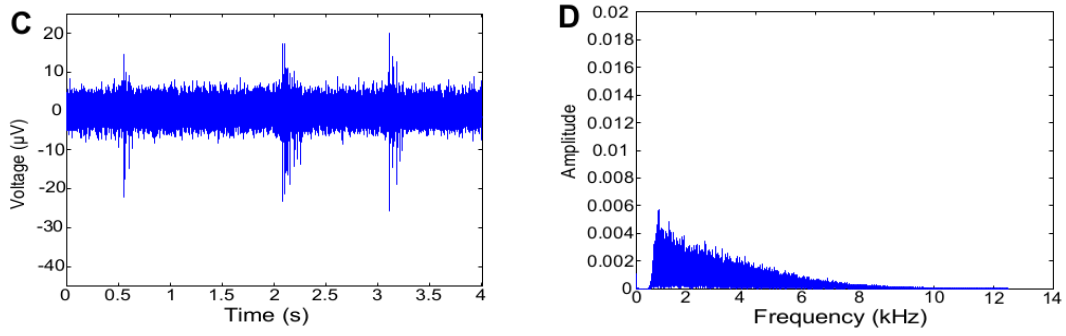

Figure 2.2: Values and spectrum of the filtered signal: C: Signal obtained after the Chebyshev type II filtering; note the stable baseline. D: Filtered signal spectrum by FFT. Slow frequency have benn suppressed from the recording.

Then, to recognize an action potential, we first calculate the **standard deviation (std)** of the recording. Then, each point overtaken three times the standard deviation is considered as an event, accounted as an action potential (see figure 2.3). We discard the events smaller length than 240 μs.

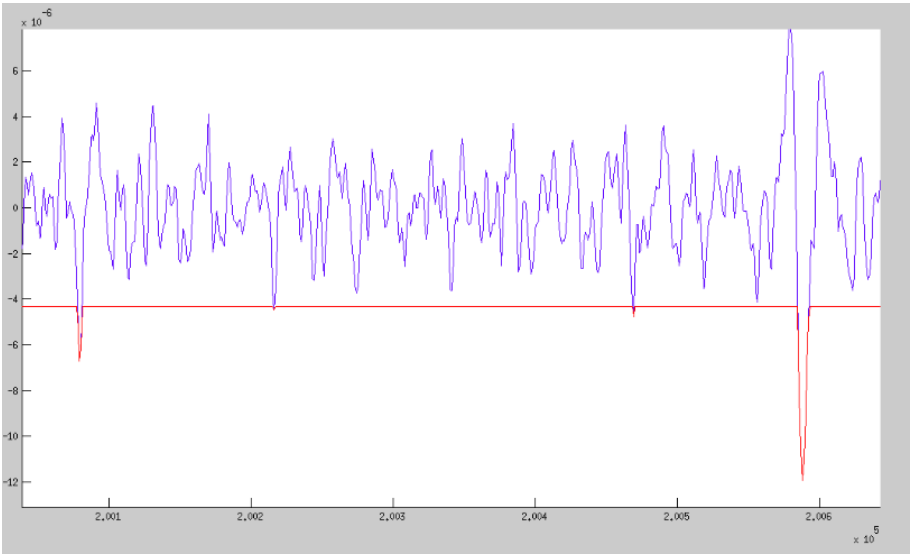

Figure 2.3: The signal (blue) and the threshold (red)

The figure 2.4 is an illustration of the extracted events, with a threshold at 2,5 times the standard deviation.

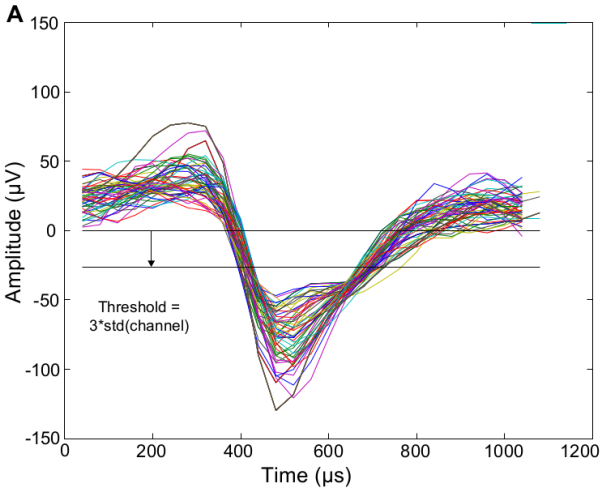

Figure 2.4: Some action potentials we extracted. The shape, kinetic and size of the events we obtained are similar to published results about a single neuron recorded by mean field, *in vivo* and *in vitro* (Berdondini 2009).

2.2.2 Results: extracted variables and events characteristics

See on the figure 2.5 the quantifications of the extracted events in a single culture. The list of all accessible features is not listed here, which could be the basis of a principal component analysis. This could be the subject of an other internship.

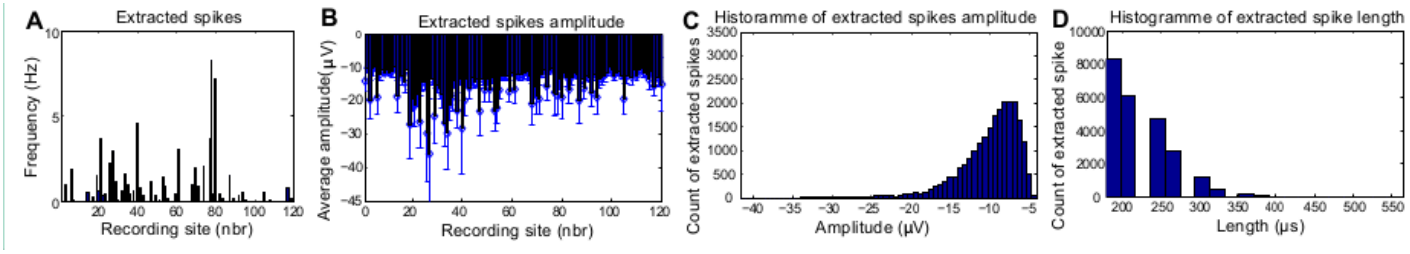

Figure 2.5: Some action potentials quantifications : **A** : Frequency of events occurence in each channels. Note that most of the channels are active, but a few of them exhibits a firing frequency higher frequency than the burst frequency. **B** : Amplitudes of the extracted events at each recording site. As previously shown, a few electrodes have a good sensibility (most probably due to the neuronal location above the electrode). **C** : Histogramm of extracted spike (action potentials) amplitudes, averaged from 6 minutes recording (2000 spikes), which indicate more than 55 spikes per seconds. **D** : Histogramm of the extracted event lenghts, which confirms that 90% of the extracted events have a length below 1,2 ms (characteristic an of action potential measured by mean field recording).

2.3 Extraction of bursts

2.3.1 The valence of the burst: a qualitative quantification of network synchronicity

At last, we studied the synchronicity of the neuronal network (detection of the bursts) by sliding a time window through the recording, and couting at each steps how many action potentials was extracted. Let us call this function the *valence* of the burst. The length of the time window we used was 100 ms.

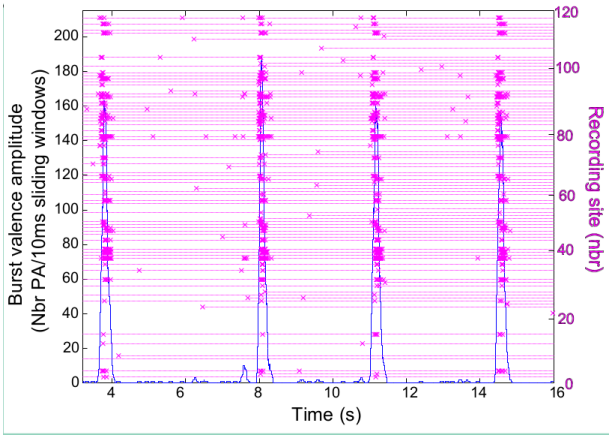

Figure 2.6: The burst valence during 16 seconds of the recording. Over, a raster plot is illustrated : a pink cross is drawn every time an action potential occurs, and each line represents a single recording site. This figures illustrates that a burst is composed of synchronical fast events of several channels.

2.3.2 Results: extracted variables and bursts characteristics

The figure 2.7 represents some quantifications we extracted from the bursts, during a 6 min recording.

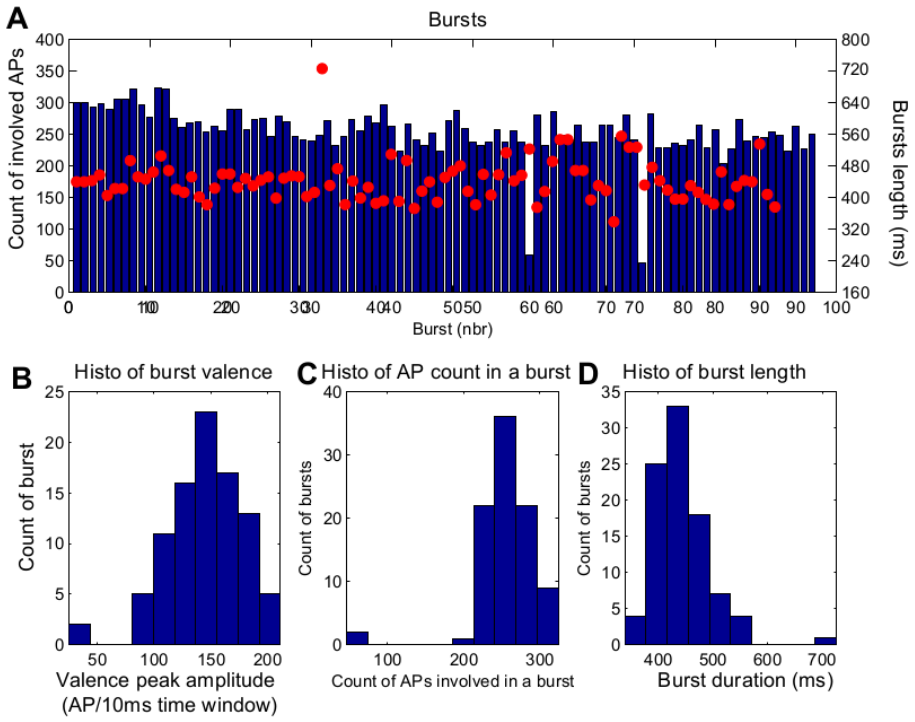

Figure 2.7: Some burst quantifications : **A** : Here is quantified (in blue) the total number of action potentials involved in a given burst. In red is illustrated the time length of each burst, which looks homogeneous. **B** : Histogramm of the burst valence amplitude. **C** : Histogramm of the number of recruited action potentials during a burst. **D** : Histogramm of the bursts duration

2.3.3 Burst patterns : can we define burst identities ?

In a culture, different bursts valences could be observed, as representated in the figure 2.8. The different shapes of burst valence with their associated rasterplots suggest two bursts fingerprints in pannels A vs. B, both exhibited by the same culture. To ensure the success of my internship, we avoid analysis of our big data set. Thus, next step of this study could be the use of strong mathematicals methods of analysis, for example the Principal Component Analysis, as previously performed in similar studies (see Gullo et al. 2006, Wanke et al. 2009)

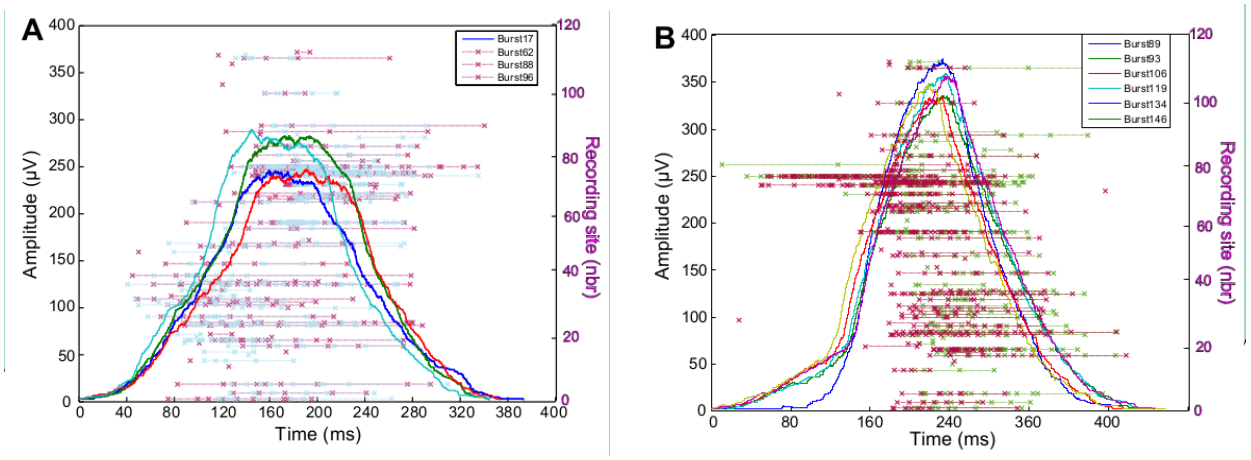

Figure 2.8: Two different burst patterns : **A** : Burst valence illustrated for four different bursts (red, green, blue and light blue). Interestingly those similarities suggests a shared «fingerprint». In addition, the raster plots of two of this bursts are overlapped in light blue and violet. It shows that the recruited channels are almost the same, while the firing pattern of each recording sites show slight differences.**B** : Same illustration for an other type of bursting pattern in the same recording. The burst valences are drawn in red, green, light blue, blue, yellow and purple, and the raster plots for two of these bursts are in yellow and red.

2.4 The Graphical User Interface (GUI)

Here are some pictures of the graphical user interface I coded.

Figure 2.9: The opening window proposing the three different operations

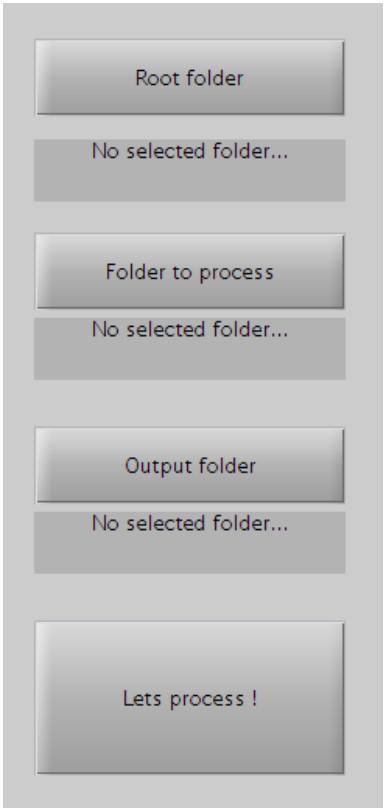

Figure 2.10: The processing window obtained by clicking on the "processing" button

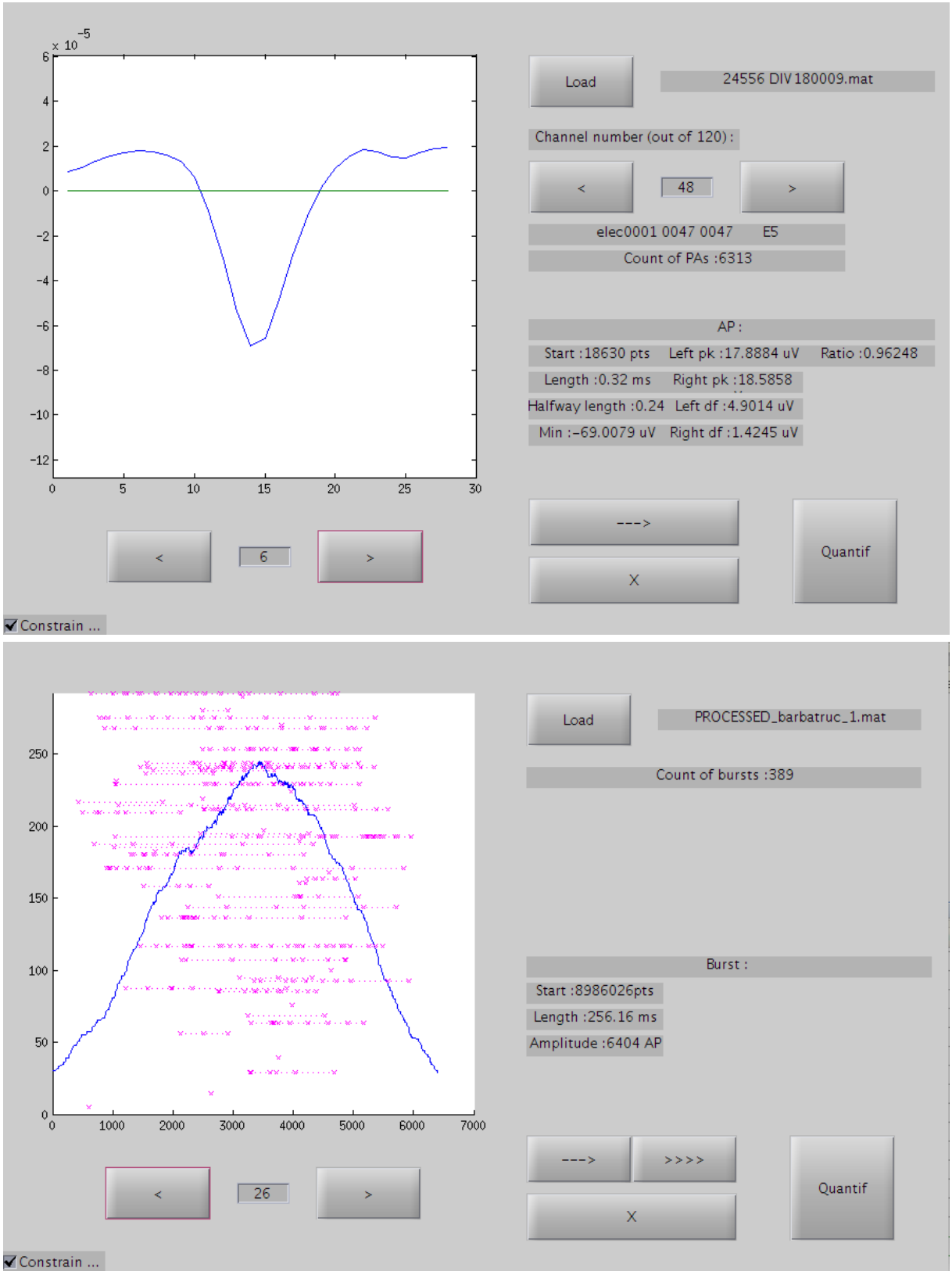

Figure 2.11: The action potentials and bursts display windows, both allowing to select a processed .mat file and to illustrate the extracted events/burst. Both windows allow singular illustration of event/burst, an overall quantification, with an possible extra window (by clicking on the arrow) to create a graphic containing choosen singular events

Conclusion

Primary neuronal cultures exhibit a strong spontaneous activity during the first two weeks *in vitro*. MEA is a suitable support to perform precise recordings of this spontaneous activity. Some extensive analysis can be done:

- single action potential isolation and quantification
- global characterization of the synchronous spontaneous activity, in the 120 channels

Therefore a more extensive study should be performed to decipher the identity of the burst types, occuring repetitively in the culture. So far, no analysis illustrates the quantity of available parameters and features that can be extracted from this neuronal network activity. Our future perspective is first to developp an automated detection and classification of these spontaneous networks rhythmic activity.

Personally, this first active experience in the research background was very interesting and intense. I was able to brush the issues, the climate, and the working conditions of this domain. I sure opened my mind to the beauty of the applied mathematics.

Chapter 3

Appendices

There follow the main part of the code written and debugged during this internship.

3.1 processing.m

```
% processing permits to extract the AP and the bursts from a .mcd file.
% processing requires the functions high_processing and burst_processing

function processing(filename, folderin, folderout)
%% Reproducing the folders tree view
ind = max(strfind(filename,'\'));
newdir = strcat(folderout, strrep(filename(1:ind), folderin, ''));
newfilename = strrep(filename, folderin, folderout); newfilename = strcat(newfilename(1:(end
    -4)), '.mat');

if exist(newfilename, 'file')==0

t1 = clock;
fprintf('\nAnalysis of %s', filename);

%% OPENING THE MCS RECORDING WITH NEUROSHARE
[~] = ns_SetLibrary('NEUROSHARE_DLL/nsMCDLibrary64.dll');
[~, hfile] = ns_OpenFile(filename);
[~, nsFileInfo] = ns_GetFileInfo(hfile);
channels_count = nsFileInfo.EntityCount;

%% EXTRACTING HIGH EVENTS
EVENTS_high = [];

fprintf('\nHigh processing... Channel :');

for i=1:channels_count
    fprintf(' %i', i);

    % Opening a channel signal
    [~, nsEntityInfo] = ns_GetEntityInfo(hfile, i);
    [~, ~, DATA] = ns_GetAnalogData(hfile, i, 1, nsEntityInfo.ItemCount);

    % Filtring the opened signal
    [~,DATA_high] = cheby_processing(DATA);

    % Adding the channel events to the final structure
    EVENTS_high = [EVENTS_high; high_processing(DATA_high, nsEntityInfo.EntityLabel)];
end

%% EXTRACTING BURST EVENTS
```

```
recording_length = nsEntityInfo.ItemCount;
EVENTS_burst = burst_processing(filename, EVENTS_high, recording_length);

%% SAVING THE STRUCTURE
if(isdir(newdir)==0)
    mkdir(newdir)
end

save(newfilename, 'filename', 'channels_count', 'recording_length', 'EVENTS_high', '
    EVENTS_burst');

fprintf('\nFile saved in %s\n', newfilename);
fprintf('Le traitement a dure %i minutes\n', etime(clock,t1)/60);
clear all;

end

end
```

3.2 cheby processing.m

```
% cheby_processing permits to filters the rawdata, thanks to an order 4 type
% II Chebyshev filter
% The output "DATA_low" is not yet implemented

function [DATA_low, DATA_high] = cheby_processing(DATA)

DATA_low = [];
[b,a] = cheby2(4, 25, 20/625, 'high');
DATA_high=filtfilt(b,a,DATA);

end
```

3.3 high processing.m

```
% high_processing permits to extract AP from a recording
% There is an error of vocab : the field "MEA" of the output structure
% should be named "channel"

function EVENTS_high = high_processing(DATA_high, channel)
%% PARAMETERS
min_event = 6;
threshold = -3.5*std(DATA_high);

%% EXTRACTING EVENTS
DATA_events = min(DATA_high, threshold*ones(size(DATA_high)))-threshold;
DATA_nonzero_ind = find(DATA_events);
DATA_interevents_time = diff(DATA_nonzero_ind);
DATA_interevents_ind = find(DATA_interevents_time > 1);
DATA_events_time = diff(DATA_interevents_ind);
DATA_events_ind = find(DATA_events_time>=min_event);
number_events = length(DATA_events_ind);

%% Filling the structure
EVENTS_high = struct('MEA', channel, 'number_events', number_events, 'start',
    DATA_nonzero_ind(DATA_interevents_ind(DATA_events_ind(1:number_events))+1)', 'length',
    DATA_events_time(DATA_events_ind(1:number_events))', 'values', {cell(1, number_events)},
```

```
    'min', [], 'peak_left', [], 'peak_right', [], 'diff_left', [], 'diff_right', [], '
    halfway_length', [], 'ratio', []);

for i = 1:number_events
    % The event duration is 10 + length under threshold + 10 (points)
    event_length = EVENTS_high.length(i);
    event_ind = (EVENTS_high.start(i)-10):(EVENTS_high.start(i)+event_length+9);
    event_values = DATA_high(event_ind);

    % Saving the events values
    EVENTS_high.values(i) = {event_values'};

    % Saving the event amplitude (minimal value under threshold)
    event_min = min(event_values(11:(10+event_length)));
    EVENTS_high.min(i) = event_min;

    % Calculation of pre and post-threshold datas : left and right peaks, left and right
        diff, halfway length and ratio
    [max_left, max_left_ind] = max(event_values(6:10));
    max_left_ind = max_left_ind + 5;
    [max_right, max_right_ind] = max(event_values((11+event_length):(15+event_length)));
    max_right_ind = max_right_ind + 10 + event_length;

    EVENTS_high.peak_left(i) = max_left;
    EVENTS_high.peak_right(i) = max_right;

    EVENTS_high.diff_left(i) = max_left - mean(event_values(1:5));
    EVENTS_high.diff_right(i) = max_right - mean(event_values((16+event_length):(20+
        event_length)));

    EVENTS_high.halfway_length(i) = length(find(event_values(max_left_ind:max_right_ind)<=(
        event_min+min(max_left, max_right))/2));
    EVENTS_high.ratio(i) = max_left/max_right;
end

end
```

3.4 burst processing.m

```
% burst_processing permits to extract the burst events from the structure EVENTS_high

function EVENTS_burst = burst_processing(filename, EVENTS_high, recording_length)

min_inter_events = 5000;
min_event_length = 10;
max_event_length = 10000;
threshold = 20; % sera modifie a la ligne 29
threshold_bas = 2;

M = 120;
N = recording_length;

pas_burst = floor(10*25000/100);

fprintf('\nBurst processing... Channel :');

Cover = zeros(1, N);
for i = 1:M
    fprintf(' %i', i);
```

```
21
22      for j=1:EVENTS_high(i).number_events
23      Cover(max((EVENTS_high(i).start(j)-pas_burst/2),1):min((EVENTS_high(i).start(j)+
            pas_burst/2), N)) =  Cover(max((EVENTS_high(i).start(j)-pas_burst/2),1):min((
            EVENTS_high(i).start(j)+pas_burst/2), N)) +1 ;
24      end
25  end
26
27  fprintf('\n');
28
29  threshold = max(10, max(Cover)/5);
30
31  %% EXTRACTING EVENTS
32  DATA_events = max(Cover, threshold);
33
34  [pks,locs] = findpeaks(DATA_events, 'MINPEAKDISTANCE', min_inter_events);
35  events_count = length(pks);
36
37  final_i = 0;
38
39  EVENTS_burst = struct('fichier', filename, 'events_count', 0, 'start', [], 'length', [], '
        peak', [], 'cover', [], 'values', []);
40
41  for i = 1:events_count
42      beginning = find(Cover(max(locs(i)-max_event_length,1):locs(i))<=threshold_bas, 1, 'last
            ');
43      ending = find(Cover(locs(i):min(locs(i)+max_event_length, N))<=threshold_bas, 1, 'first'
            );
44
45      if or(isempty(beginning), isempty(ending))
46      elseif (locs(i) + ending - max(locs(i)-max_event_length,1) + beginning) >=
            min_event_length %length of any burst mut be >= min_event_length
47          final_i = final_i + 1;
48          EVENTS_burst(final_i).peak = pks(i);
49          EVENTS_burst(final_i).start = max(locs(i)-max_event_length,1) + beginning;
50          EVENTS_burst(final_i).length = locs(i) + ending - EVENTS_burst(final_i).start;
51          EVENTS_burst(final_i).cover = Cover(EVENTS_burst(final_i).start:(EVENTS_burst(
                final_i).start+EVENTS_burst(final_i).length-1));
52          EVENTS_burst(final_i).values = cell(120, 1);
53          for j = 1:120
54              ind = find(and(EVENTS_high(j).start >= EVENTS_burst(final_i).start, EVENTS_high(
                    j).start <= (EVENTS_burst(final_i).start + EVENTS_burst(final_i).length)));
55              EVENTS_burst(final_i).values(j) = {EVENTS_high(j).start(ind)};
56          end
57      end
58
59  end
60
61  EVENTS_burst(1).events_count = final_i;
62
63  end
```

3.5 visual.m

```
1  % visual is a GUI which permits to process and display datas from .mcd
2  % be careful while closing a window, it could makes visual crash
3
4  function visual
5  %% CONTROL MENU
```

```
6
7  hfig_menu = figure('Visible', 'On', 'Position', [360 500 250 400]);
8
9  hbutton_menu_proc = uicontrol('Style', 'pushbutton', ...
10                       'String', 'PROCESSING', 'Position', [25 275 200 100], ...
11                       'Callback', {@hbutton_menu_proc_Callback}, ...
12                       'Parent', hfig_menu);
13
14 hbutton_menu_visual = uicontrol('Style', 'pushbutton', ...
15                       'String', 'VISUAL', 'Position', [25 150 200 100], ...
16                       'Callback', {@hbutton_menu_visual_Callback}, ...
17                       'Parent', hfig_menu);
18
19 hbutton__menu_cat = uicontrol('Style', 'pushbutton', ...
20                       'String', 'CAT', 'Position', [25 25 200 100], ...
21                       'Callback', {@hbutton_menu_cat_Callback}, ...
22                       'Parent', hfig_menu);
23
24 hbutton_menu_proc_record = uicontrol('Style', 'pushbutton', 'Visible', 'Off', ...
25                       'String', 'RECORD', 'Position', [25 275 85 100], ...
26                       'Callback', {@hbutton_menu_proc_record_Callback}, ...
27                       'Parent', hfig_menu);
28
29 hbutton_menu_proc_routine = uicontrol('Style', 'pushbutton', 'Visible', 'Off', ...
30                       'String', 'ROUTINE', 'Position', [140 275 85 100], ...
31                       'Callback', {@hbutton_menu_proc_routine_Callback}, ...
32                       'Parent', hfig_menu);
33
34 hbutton_menu_visual_AP = uicontrol('Style', 'pushbutton', 'Visible', 'Off', ...
35                       'String', 'AP', 'Position', [25 150 85 100], ...
36                       'Callback', {@hbutton_menu_visual_AP_Callback}, ...
37                       'Parent', hfig_menu);
38
39 hbutton_menu_visual_burst = uicontrol('Style', 'pushbutton', 'Visible', 'Off', ...
40                       'String', 'BURST', 'Position', [140 150 85 100], ...
41                       'Callback', {@hbutton_menu_visual_burst_Callback}, ...
42                       'Parent', hfig_menu);
43
44 set(hfig_menu, 'Name', 'MENU');
45 movegui(hfig_menu, 'center');
46
47 %% CONTROL PROC_RECORD
48
49 hfig_record = figure('Visible', 'Off', 'Position', [360 500 250 500]);
50
51 hbutton_record_files = uicontrol('Style', 'pushbutton', ...
52                       'String', 'Files to process', 'Position', [25 425 200 50], ...
53                       'Callback', {@hbutton_record_files_Callback}, ...
54                       'Parent', hfig_record);
55
56 hbutton_record_folderout = uicontrol('Style', 'pushbutton', ...
57                       'String', 'Output folder', 'Position', [25 200 200 50], ...
58                       'Callback', {@hbutton_record_folderout_Callback}, ...
59                       'Parent', hfig_record);
60
61 hbutton_record_proc = uicontrol('Style', 'pushbutton', ...
62                       'String', 'Lets process !', 'Position', [25 25 200 100], ...
63                       'Callback', {@hbutton_record_proc_Callback}, ...
64                       'Parent', hfig_record);
65
```

```
66 htext_record_files = uicontrol('Style', 'text', ...
67                          'String', 'No selected files...', ...
68                          'Position', [25 280 200 140], ...
69                          'Parent', hfig_record);
70
71 htext_record_folderout = uicontrol('Style', 'text', ...
72                          'String', 'No selected folder...', ...
73                          'Position', [25 155 200 40], ...
74                          'Parent', hfig_record);
75
76 movegui(hfig_record, 'center');
77
78 %% CONTROL PROC_ROUTINE
79
80 hfig_routine = figure('Visible', 'Off', 'Position', [360 500 250 525]);
81
82 hbutton_routine_folderin = uicontrol('Style', 'pushbutton', ...
83                     'String', 'Root folder', 'Position', [25 450 200 50], ...
84                     'Callback', {@hbutton_routine_folderin_Callback}, ...
85                     'Parent', hfig_routine);
86
87 hbutton_routine_folder = uicontrol('Style', 'pushbutton', ...
88                     'String', 'Folder to process', 'Position', [25 325 200 50], ...
89                     'Callback', {@hbutton_routine_folder_Callback}, ...
90                     'Parent', hfig_routine);
91
92 hbutton_routine_folderout = uicontrol('Style', 'pushbutton', ...
93                     'String', 'Output folder', 'Position', [25 200 200 50], ...
94                     'Callback', {@hbutton_routine_folderout_Callback}, ...
95                     'Parent', hfig_routine);
96
97 hbutton_routine_proc = uicontrol('Style', 'pushbutton', ...
98                     'String', 'Lets process !', 'Position', [25 25 200 100], ...
99                     'Callback', {@hbutton_routine_proc_Callback}, ...
100                    'Parent', hfig_routine);
101
102 htext_routine_folderin = uicontrol('Style', 'text', ...
103                          'String', 'No selected folder...', ...
104                          'Position', [25 395 200 40], ...
105                          'Parent', hfig_routine);
106
107 htext_routine_folder = uicontrol('Style', 'text', ...
108                          'String', 'No selected folder...', ...
109                          'Position', [25 280 200 40], ...
110                          'Parent', hfig_routine);
111
112 htext_routine_folderout = uicontrol('Style', 'text', ...
113                          'String', 'No selected folder...', ...
114                          'Position', [25 155 200 40], ...
115                          'Parent', hfig_routine);
116
117 set(hfig_routine, 'Name', 'ROUTINE');
118 movegui(hfig_routine, 'center');
119
120 %% CONTROL VISUAL_AP
121
122 hfig_AP = figure('Visible', 'Off', 'Position', [360 500 900 600]);
123 ha_AP = axes('Units', 'pixels', 'Position', [50 150 400 400], 'Parent', hfig_AP);
124
125 hfig_AP_multi = figure('Visible', 'Off');
```

```
126 ha_AP_multi = axes('Parent', hfig_AP_multi);
127
128 hbutton_AP_go = uicontrol('Visible', 'Off', 'Style', 'pushbutton', ...
129                       'String', '--->', 'Position', [500 85 200 45], ...
130                       'Callback', {@hbutton_AP_go_Callback}, ...
131                       'Parent', hfig_AP);
132
133 hbutton_AP_remove = uicontrol('Visible', 'Off', 'Style', 'pushbutton', ...
134                       'String', 'X', 'Position', [500 30 200 45], ...
135                       'Callback', {@hbutton_AP_remove_Callback}, ...
136                       'Parent', hfig_AP);
137
138 hbutton_AP_left = uicontrol('Visible', 'Off', 'Style', 'pushbutton', ...
139                       'String', '<', 'Position', [100 50 100 50], ...
140                       'Callback', {@hbutton_AP_left_Callback}, ...
141                       'Parent', hfig_AP);
142
143 hbutton_AP_right = uicontrol('Visible', 'Off', 'Style', 'pushbutton', ...
144                       'String', '>', 'Position', [300 50 100 50], ...
145                       'Callback', {@hbutton_AP_right_Callback}, ...
146                       'Parent', hfig_AP);
147
148 hbutton_channel_left = uicontrol('Visible', 'Off', 'Style', 'pushbutton', ...
149                       'String', '<', 'Position', [500 400 100 50], ...
150                       'Callback', {@hbutton_channel_left_Callback}, ...
151                       'Parent', hfig_AP);
152
153 hbutton_channel_right = uicontrol('Visible', 'Off', 'Style', 'pushbutton', ...
154                       'String', '>', 'Position', [700 400 100 50], ...
155                       'Callback', {@hbutton_channel_right_Callback}, ...
156                       'Parent', hfig_AP);
157
158 hbutton_AP_quantif = uicontrol('Visible', 'Off', 'Style', 'pushbutton', ...
159                       'String', 'Quantif', 'Position', [750 85 100 45], ...
160                       'Callback', {@hbutton_AP_quantif_Callback}, ...
161                       'Parent', hfig_AP);
162 hbutton_AP_rasterplot = uicontrol('Visible', 'Off', 'Style', 'pushbutton', ...
163                       'String', 'Rasterplot', 'Position', [750 30 100 45], ...
164                       'Callback', {@hbutton_AP_rasterplot_Callback}, ...
165                       'Parent', hfig_AP);
166
167 hbutton_AP_load = uicontrol('Style', 'pushbutton', ...
168                       'String', 'Load', 'Position', [500 500 100 50], ...
169                       'Callback', {@hbutton_AP_load_Callback}, ...
170                       'Parent', hfig_AP);
171
172 hedit_channel = uicontrol('Visible', 'Off', 'Style', 'edit', ...
173                     'String', '0', ...
174                     'Position', [625 415 50 20], ...
175                     'Callback', {@hedit_channel_Callback}, ...
176                     'Parent', hfig_AP);
177
178 hedit_AP = uicontrol('Visible', 'Off', 'Style', 'edit', ...
179                     'String', '0', ...
180                     'Position', [225 65 50 20], ...
181                     'Callback', {@hedit_AP_Callback}, ...
182                     'Parent', hfig_AP);
183
184 hcb_AP = uicontrol('Style', 'checkbox', 'Visible', 'Off',...
185                   'String', 'Constrain_axes',...
```

```
                'Value', 1, 'Position',[0 0 100 20], ...
                'Callback', {@hcb_AP_Callback}, ...
                'Parent', hfig_AP);

htext_channel = uicontrol('Visible', 'Off', 'Style', 'text', ...
                          'String', 'Channel number (out of 120) :', ...
                          'Position', [500 460 200 20], ...
                          'Parent', hfig_AP);

htext_channel_name = uicontrol('Visible', 'Off', 'Style', 'text', ...
                          'String', '', ...
                          'Position', [500 370 300 20], ...
                          'Parent', hfig_AP);

htext_channel_data = uicontrol('Visible', 'Off', 'Style', 'text', ...
                          'String', '', ...
                          'Position', [500 345 300 20], ...
                          'Parent', hfig_AP);

htext_AP_data = uicontrol('Visible', 'Off', 'Style', 'text', ...
                          'String', 'AP : ', ...
                          'Position', [500 280 385 20], ...
                          'Parent', hfig_AP);

htext_AP_start = uicontrol('Visible', 'Off', 'Style', 'text', ...
                          'String', '', ...
                          'Position', [500 255 130 20], ...
                          'Parent', hfig_AP);

htext_AP_length = uicontrol('Visible', 'Off', 'Style', 'text', ...
                          'String', '', ...
                          'Position', [500 230 130 20], ...
                          'Parent', hfig_AP);

htext_AP_halfway_length = uicontrol('Visible', 'Off', 'Style', 'text', ...
                          'String', '', ...
                          'Position', [500 205 130 20], ...
                          'Parent', hfig_AP);

htext_AP_min = uicontrol('Visible', 'Off', 'Style', 'text', ...
                          'String', '', ...
                          'Position', [500 180 130 20], ...
                          'Parent', hfig_AP);

htext_AP_peak_left = uicontrol('Visible', 'Off', 'Style', 'text', ...
                          'String', '', ...
                          'Position', [630 255 130 20], ...
                          'Parent', hfig_AP);

htext_AP_peak_right = uicontrol('Visible', 'Off', 'Style', 'text', ...
                          'String', '', ...
                          'Position', [630 230 130 20], ...
                          'Parent', hfig_AP);

htext_AP_diff_left = uicontrol('Visible', 'Off', 'Style', 'text', ...
                          'String', '', ...
                          'Position', [630 205 130 20], ...
                          'Parent', hfig_AP);

htext_AP_diff_right = uicontrol('Visible', 'Off', 'Style', 'text', ...
```

```
                            'String', '', ...
                            'Position', [630 180 130 20], ...
                            'Parent', hfig_AP);
htext_AP_ratio = uicontrol('Visible', 'Off', 'Style', 'text', ...
                            'String', '', ...
                            'Position', [760 255 125 20], ...
                            'Parent', hfig_AP);

htext_file = uicontrol('Style', 'text', 'String', 'No loaded file...', ...
                            'Position', [625 515 260 20],...
                            'Parent', hfig_AP);

set(hfig_AP, 'Name', 'VISUAL_AP');
movegui(hfig_AP, 'center');
set(hfig_AP, 'toolbar', 'figure');


%% CONTROL VISUAL_BURST

hfig_burst = figure('Visible', 'Off', 'Position', [360 500 900 600]);
hfig_burst_multi = figure('Visible', 'Off');

ha_burst = axes('Units', 'pixels', 'Position', [50 150 400 400], 'Parent', hfig_burst);
ha_burst_multi = axes('Parent', hfig_burst_multi);

hold(ha_burst_multi, 'on');

hbutton_burst_go = uicontrol('Visible', 'Off', 'Style', 'pushbutton', ...
                    'String', '--->', 'Position', [500 85 100 45], ...
                    'Callback', {@hbutton_burst_go_Callback}, ...
                    'Parent', hfig_burst);

hbutton_burst_go2 = uicontrol('Visible', 'Off', 'Style', 'pushbutton', ...
                    'String', '>>>>', 'Position', [600 85 100 45], ...
                    'Callback', {@hbutton_burst_go2_Callback}, ...
                    'Parent', hfig_burst);

hbutton_burst_remove = uicontrol('Visible', 'Off', 'Style', 'pushbutton', ...
                    'String', 'X', 'Position', [500 30 200 45], ...
                    'Callback', {@hbutton_burst_remove_Callback}, ...
                    'Parent', hfig_burst);

hbutton_burst_left = uicontrol('Visible', 'Off', 'Style', 'pushbutton', ...
                    'String', '<', 'Position', [100 50 100 50], ...
                    'Callback', {@hbutton_burst_left_Callback}, ...
                    'Parent', hfig_burst);

hbutton_burst_right = uicontrol('Visible', 'Off', 'Style', 'pushbutton', ...
                    'String', '>', 'Position', [300 50 100 50], ...
                    'Callback', {@hbutton_burst_right_Callback}, ...
                    'Parent', hfig_burst);

hbutton_burst_load = uicontrol('Style', 'pushbutton', ...
                    'String', 'Load', 'Position', [500 500 100 50], ...
                    'Callback', {@hbutton_burst_load_Callback}, ...
                    'Parent', hfig_burst);

hbutton_burst_quantif = uicontrol('Visible', 'Off', 'Style', 'pushbutton', ...
                    'String', 'Quantif', 'Position', [750 30 100 100], ...
                    'Callback', {@hbutton_burst_quantif_Callback}, ...
```

```
306                     'Parent', hfig_burst);
307
308 hcb_burst = uicontrol('Style', 'checkbox', 'Visible', 'Off',...
309                 'String', 'Constrain_axes',...
310                 'Value', 1, 'Position',[0 0 100 20], ...
311                 'Callback', {@hcb_burst_Callback}, ...
312                 'Parent', hfig_burst);
313
314 hedit_burst = uicontrol('Visible', 'Off', 'Style', 'edit', ...
315                   'String', '0', ...
316                   'Position', [225 65 50 20], ...
317                   'Callback', {@hedit_burst_Callback}, ...
318                   'Parent', hfig_burst);
319
320 htext_burst_file = uicontrol('Style', 'text', ...
321                     'String', 'No_loaded_file...', ...
322                     'Position', [625 515 250 20], ...
323                     'Parent', hfig_burst);
324
325 htext_file_data = uicontrol('Style', 'text', 'Visible', 'Off',...
326                     'String', '0', ...
327                     'Position', [500 460 375 20], ...
328                     'Parent', hfig_burst);
329
330 htext_burst_data = uicontrol('Visible', 'Off', 'Style', 'text', ...
331                             'String', 'Burst_:_', ...
332                             'Position', [500 280 385 20], ...
333                             'Parent', hfig_burst);
334
335 htext_burst_start = uicontrol('Visible', 'Off', 'Style', 'text', ...
336                             'String', '', ...
337                             'Position', [500 255 130 20], ...
338                             'Parent', hfig_burst);
339
340 htext_burst_length = uicontrol('Visible', 'Off', 'Style', 'text', ...
341                             'String', '', ...
342                             'Position', [500 230 130 20], ...
343                             'Parent', hfig_burst);
344
345 htext_burst_max = uicontrol('Visible', 'Off', 'Style', 'text', ...
346                             'String', '', ...
347                             'Position', [500 205 130 20], ...
348                             'Parent', hfig_burst);
349
350 htext_burst_AP_count = uicontrol('Visible', 'Off', 'Style', 'text', ...
351                             'String', '', ...
352                             'Position', [500 180 130 20], ...
353                             'Parent', hfig_burst);
354
355 set(hfig_burst, 'Name', 'VISUAL_BURST');
356 movegui(hfig_burst, 'center');
357 set(hfig_burst, 'toolbar', 'figure');
358
359
360 %% CONTROL CAT
361
362 hfig_cat = figure('Visible', 'Off', 'Position', [360 500 250 625]);
363
364 hfig_cat_proc = figure('Visible', 'Off', 'Position', [360 500 250 400]);
365
```

```
366 hbutton_cat_load = uicontrol('Style', 'pushbutton', ...
367                     'String', 'Load', 'Position', [25 500 200 100], ...
368                     'Callback', {@hbutton_cat_load_Callback}, ...
369                     'Parent', hfig_cat);
370
371 hbutton_cat_cut = uicontrol('Style', 'pushbutton', ...
372                     'String', 'Cut', 'Position', [150 0 100 100], ...
373                     'Callback', {@hbutton_cat_cut_Callback}, ...
374                     'Parent', hfig_cat_proc);
375
376 hbutton_cat_continue = uicontrol('Style', 'pushbutton', ...
377                     'String', 'Dont␣cut', 'Position', [150 110 100 100], ...
378                     'Callback', {@hbutton_cat_continue_Callback}, ...
379                     'Parent', hfig_cat_proc);
380
381 hbutton_cat_plot = uicontrol('Style', 'pushbutton', ...
382                     'String', 'Rasterplot', 'Position', [25 250 200 75], ...
383                     'Callback', {@hbutton_cat_plot_Callback}, ...
384                     'Parent', hfig_cat_proc);
385
386 hedit_cat_left = uicontrol('Visible', 'On', 'Style', 'edit', ...
387                            'String', '0', ...
388                            'Position', [0 25 60 20], ...
389                            'Parent', hfig_cat_proc);
390
391 hedit_cat_right = uicontrol('Visible', 'On', 'Style', 'edit', ...
392                            'String', '0', ...
393                            'Position', [80 25 60 20], ...
394                            'Parent', hfig_cat_proc);
395
396 htext_cat_state = uicontrol('Visible', 'Off', 'Style', 'text', ...
397                            'String', '', ...
398                            'Position', [0 240 250 20], ...
399                            'Parent', hfig_cat);
400
401 htext_cat_load = uicontrol('Visible', 'On', 'Style', 'text', ...
402                            'String', 'No␣loaded␣files', ...
403                            'Position', [0 280 250 200], ...
404                            'Parent', hfig_cat);
405
406 htext_cat_new = uicontrol('Visible', 'Off', 'Style', 'text', ...
407                            'String', 'No␣loaded␣files', ...
408                            'Position', [0 20 250 200], ...
409                            'Parent', hfig_cat);
410
411 htext_cat_step = uicontrol('Visible', 'On', 'Style', 'text', ...
412                            'String', '', ...
413                            'Position', [0 375 250 20], ...
414                            'Parent', hfig_cat_proc);
415
416 htext_cat_file = uicontrol('Visible', 'On', 'Style', 'text', ...
417                            'String', '', ...
418                            'Position', [0 350 250 20], ...
419                            'Parent', hfig_cat_proc);
420
421 htext_cat_left = uicontrol('Visible', 'On', 'Style', 'text', ...
422                            'String', 'Left␣cut', ...
423                            'Position', [0 55 60 20], ...
424                            'Parent', hfig_cat_proc);
425
```

```
426 htext_cat_right = uicontrol('Visible', 'On', 'Style', 'text', ...
427                             'String', 'Right_cut', ...
428                             'Position', [80 55 60 20], ...
429                             'Parent', hfig_cat_proc);
430
431 movegui(hfig_cat, 'center');
432 %% BODY
433
434 % MENU's variables
435 folderin = '';
436 folderout = '';
437 folder = '';
438
439 % RECORD's variables
440 record_filename = '';
441 record_pathname = '';
442 record_folderout = '';
443 record_files_count = '';
444
445 % AP's variables
446 i_AP = 1;
447 AP_filename = '';
448 AP_pathname = '';
449 AP_fullfilename = '';
450 AP_data_struct = [];
451 AP_data = [];
452 AP_multi_values = [];
453 AP_multi_legend = cell(0);
454 AP_ymin = 0;
455 AP_ymax = 0;
456 events_count = 1;
457 i_channel = 1;
458
459 % BURST's variables
460 i_burst = 1;
461 burst_filename = '';
462 burst_pathname = '';
463 burst_fullfilename = '';
464 bursts_count = 1;
465 burst_data_struct = [];
466 burst_data = [];
467 burst_ymax = 0;
468 burst_multi_values = [];
469 burst_multi_legend = cell(0);
470
471 % CAT's variables
472 files_count = 0;
473 i_cat = 1;
474 n_cat = 1;
475 M_cat = [];
476 files = [];
477 time = 0;
478 filenames = [];
479 cat_pathname = '';
480 newfilenames = [];
481
482
483 %% CALLBACKS MENU
484
485     function hbutton_menu_proc_Callback(~, ~)
```

```
        set(hbutton_menu_proc, 'Visible', 'Off');
        set(hbutton_menu_proc_record, 'Visible', 'On');
        set(hbutton_menu_proc_routine, 'Visible', 'On');
    end

    function hbutton_menu_visual_Callback(~, ~)
        set(hbutton_menu_visual, 'Visible', 'Off');
        set(hbutton_menu_visual_AP, 'Visible', 'On');
        set(hbutton_menu_visual_burst, 'Visible', 'On');
    end

    function hbutton_menu_cat_Callback(~, ~)
        set(hfig_menu, 'Visible', 'Off');
        set(hfig_cat, 'Visible', 'On');
    end

    function hbutton_menu_proc_record_Callback(~, ~)
        set(hfig_menu, 'Visible', 'Off');
        set(hfig_record, 'Visible', 'On');
    end

    function hbutton_menu_proc_routine_Callback(~, ~)
        set(hfig_menu, 'Visible', 'Off');
        set(hfig_routine, 'Visible', 'On')
    end

    function hbutton_menu_visual_AP_Callback(~, ~)
        set(hfig_menu, 'Visible', 'Off');
        set(hfig_AP, 'Visible', 'On');
    end

    function hbutton_menu_visual_burst_Callback(~, ~)
        set(hfig_menu, 'Visible', 'Off');
        set(hfig_burst, 'Visible', 'On')
    end

%% CALLBACKS PROC_RECORD

    function  hbutton_record_files_Callback(~, ~)
        set(htext_record_files, 'String', '');

        [record_filename,record_pathname,filterindex] = uigetfile({'*.mcd', 'MCD Files (*.
            mcd)'}, ...
                                                                  'Pick some mcd files', '
                                                                      MultiSelect','On');

        if filterindex
            record_files_count = length(record_filename);
            set(htext_record_files, 'String', record_filename)
        end
    end

    function hbutton_record_folderout_Callback(~, ~)
        record_folderout = uigetdir('C:\Users\raphael\Desktop');
        set(htext_record_folderout, 'String', record_folderout);
    end

    function hbutton_record_proc_Callback(~, ~)
            set(hbutton_record_proc, 'String', 'Processing...');
            if iscell(record_filename)
```

```
544                 for i =1:record_files_count
545                     processing(strcat(record_pathname, record_filename{i}), strcat(
                            record_pathname), strcat(record_folderout, '\'));
546                 end
547             else
548                 processing(strcat(record_pathname, record_filename), strcat(record_pathname)
                        , strcat(record_folderout, '\'));
549             end
550             set(hbutton_record_proc, 'String', 'Processed ! Click to process again');
551     end
552
553 %% CALLBACKS PROC_ROUTINE
554
555     function hbutton_routine_folderin_Callback(~, ~)
556         folderin = uigetdir('C:\');
557         set(htext_routine_folderin, 'String', folderin);
558     end
559
560     function hbutton_routine_folder_Callback(~, ~)
561         if isempty(folderin)
562             folder = uigetdir('C:\');
563         else
564             folder = uigetdir(strcat(folderin, '\'));
565         end
566         set(htext_routine_folder, 'String', folder);
567     end
568
569     function hbutton_routine_folderout_Callback(~, ~)
570         folderout = uigetdir('C:\Users\raphael\Desktop');
571         set(htext_routine_folderout, 'String', folderout);
572     end
573
574     function hbutton_routine_proc_Callback(~, ~)
575         set(hbutton_routine_proc, 'String', 'Processing...');
576         routine(strcat(folder, '\'), strcat(folderin, '\'), strcat(folderout, '\'));
577         set(hbutton_routine_proc, 'String', 'Processed ! Click to process again');
578     end
579
580
581 %% CALLBACKS VISUAL_AP
582
583     function hbutton_AP_left_Callback(~, ~)
584         i_AP = max(1, i_AP-1);
585         set(hedit_AP, 'String', int2str(i_AP));
586         hedit_AP_Callback;
587     end
588
589     function hbutton_AP_right_Callback(~, ~)
590         i_AP=min(i_AP+1, events_count);
591         set(hedit_AP, 'String', int2str(i_AP));
592         hedit_AP_Callback;
593     end
594
595     function hbutton_channel_left_Callback(~, ~)
596         i_channel =i_channel-1;
597         set(hedit_channel, 'String', int2str(i_channel));
598         hedit_channel_Callback;
599     end
600
601     function hbutton_channel_right_Callback(~, ~)
```

```
602         i_channel = i_channel+1;
603         set(hedit_channel, 'String', int2str(i_channel));
604         hedit_channel_Callback;
605     end
606 
607     function hbutton_AP_load_Callback(~, ~)
608         i_AP = 0;
609         i_channel = 0;
610         cla(ha_AP);
611         set(hedit_AP, 'String', '0');
612         set(hbutton_AP_left, 'Visible', 'Off');
613         set(hbutton_AP_right, 'Visible', 'Off');
614         set(hbutton_channel_left, 'Visible', 'Off');
615         set(hbutton_channel_right, 'Visible', 'Off');
616         set(hedit_AP, 'Visible', 'Off');
617         set(htext_channel_name, 'Visible', 'Off');
618         set(htext_channel_data, 'Visible', 'Off');
619         set(htext_channel, 'Visible', 'Off')
620         set(hedit_channel, 'Visible', 'Off');
621         set(hedit_channel, 'String', '0');
622         set(hbutton_AP_go, 'Visible', 'Off');
623         set(hbutton_AP_remove, 'Visible', 'Off');
624         set(htext_file_data, 'Visible', 'Off');
625         set(hbutton_AP_quantif, 'Visible', 'Off');
626         set(hbutton_AP_rasterplot, 'Visible', 'Off');
627         set(hcb_AP, 'Visible', 'Off');
628         set(htext_AP_data, 'Visible', 'Off');
629         set(htext_AP_start, 'Visible', 'Off');
630         set(htext_AP_length, 'Visible', 'Off');
631         set(htext_AP_halfway_length, 'Visible', 'Off');
632         set(htext_AP_min, 'Visible', 'Off');
633         set(htext_AP_peak_left, 'Visible', 'Off');
634         set(htext_AP_peak_right, 'Visible', 'Off');
635         set(htext_AP_diff_left, 'Visible', 'Off');
636         set(htext_AP_diff_right, 'Visible', 'Off');
637         set(htext_AP_ratio, 'Visible', 'Off');
638 
639         [AP_filename, AP_pathname, filterindex] = uigetfile({'*.m;*.mat', 'MATLAB␣Files␣(*.m
                ,␣*.mat)'}, ...
640                                                             'Pick␣a␣file␣containing␣
                                                                EVENTS_high');
641 
642         if filterindex
643             AP_fullfilename = strcat(AP_pathname, AP_filename);
644             AP_data_struct = load(AP_fullfilename);
645 
646             set(htext_file, 'String', AP_filename);
647             set(hedit_channel, 'Visible', 'On');
648             set(htext_channel, 'Visible', 'On');
649             set(hbutton_channel_left, 'Visible', 'On');
650             set(hbutton_channel_right, 'Visible', 'On');
651             set(hbutton_AP_quantif, 'Visible', 'On');
652             set(hbutton_AP_rasterplot, 'Visible', 'On');
653             set(hcb_AP, 'Visible', 'On');
654 
655             [mini, maxi] = cellfun(@AP_minmax, {AP_data_struct.EVENTS_high.values}, '
                    UniformOutput', 0);
656             AP_ymin = min(cell2mat(mini)); AP_ymax = max(cell2mat(maxi));
657 
658             hedit_channel_Callback;
```

```
        else
            set(htext_file, 'String', 'No loaded file...');
        end
    end

    function hedit_channel_Callback(~, ~)
        i_AP = 0;
        i_channel = min(max(str2num(get(hedit_channel, 'String')), 1), 120);
        AP_data = AP_data_struct.EVENTS_high(i_channel);
        events_count = AP_data.number_events;

        if events_count>0
            i_AP = 1;
            set(htext_AP_data, 'Visible', 'On');
            set(hbutton_AP_left, 'Visible', 'On');
            set(hbutton_AP_right, 'Visible', 'On');
            set(hbutton_AP_go, 'Visible', 'On');
            set(hbutton_AP_remove, 'Visible', 'On');
            set(hedit_AP, 'String', '1');
            hedit_AP_Callback;
        else
            set(htext_AP_data, 'Visible', 'Off');
            set(htext_AP_start, 'Visible', 'Off');
            set(htext_AP_length, 'Visible', 'Off');
            set(htext_AP_halfway_length, 'Visible', 'Off');
            set(htext_AP_min, 'Visible', 'Off');
            set(htext_AP_peak_left, 'Visible', 'Off');
            set(htext_AP_peak_right, 'Visible', 'Off');
            set(htext_AP_diff_left, 'Visible', 'Off');
            set(htext_AP_diff_right, 'Visible', 'Off');
            set(htext_AP_ratio, 'Visible', 'Off');

            set(hbutton_AP_left, 'Visible', 'Off');
            set(hbutton_AP_right, 'Visible', 'Off');
            set(hbutton_AP_go, 'Visible', 'Off');
            set(hbutton_AP_remove, 'Visible', 'Off');
            cla(ha_AP);
        end

        set(htext_channel_name, 'String', AP_data.MEA, 'Visible', 'On');
        txt = strcat('Count of PAs : ', int2str(events_count));
        set(htext_channel_data, 'String', txt, 'Visible', 'On');
        set(hedit_channel, 'String', int2str(i_channel));
        set(hedit_AP, 'Visible', 'On');
    end

    function hedit_AP_Callback(~, ~)
        i_AP = min(max(str2num(get(hedit_AP, 'String')), 1), events_count);

        plot(ha_AP, [cell2mat(AP_data.values(i_AP)); zeros(size(cell2mat(AP_data.values(i_AP
            ))))]');
        if get(hcb_AP, 'Value')
            set(ha_AP, 'YLim', [AP_ymin, AP_ymax]);
        end

        set(hedit_AP, 'String', int2str(i_AP));
        set(htext_AP_start, 'Visible', 'On', 'String', strcat('Start : ', int2str(AP_data.
            start(i_AP)), ' pts'));
        set(htext_AP_length, 'Visible', 'On', 'String', strcat('Length : ', num2str(AP_data.
            length(i_AP)/25000*1000), ' ms'));
```

```
716         set(htext_AP_halfway_length, 'Visible', 'On', 'String', strcat('Halfway length : ',
                num2str(AP_data.halfway_length(i_AP)/25000*1000), ' ms'));
717         set(htext_AP_min, 'Visible', 'On', 'String', strcat('Min : ', num2str(AP_data.min(
                i_AP)*1000000), ' uV'));
718         set(htext_AP_peak_left, 'Visible', 'On', 'String', strcat('Left peak : ', num2str(
                AP_data.peak_left(i_AP)*1000000), ' uV'));
719         set(htext_AP_peak_right, 'Visible', 'On', 'String', strcat('Right peak : ', num2str(
                AP_data.peak_right(i_AP)*1000000), ' uV'));
720         set(htext_AP_diff_left, 'Visible', 'On', 'String', strcat('Left diff : ', num2str(
                AP_data.diff_left(i_AP)*1000000), ' uV'));
721         set(htext_AP_diff_right, 'Visible', 'On', 'String', strcat('Right diff : ', num2str(
                AP_data.diff_right(i_AP)*1000000), ' uV'));
722         set(htext_AP_ratio, 'Visible', 'On', 'String', strcat('Ratio : ', num2str(AP_data.
                ratio(i_AP))));
723     end
724
725     function hbutton_AP_go_Callback(~,~)
726         newAP = cell2mat(AP_data.values(i_AP))';
727         [L1, l] = size(AP_multi_values);
728         L2 = length(newAP);
729
730         if L1 > L2
731                 newAP = [newAP; nan(L1-L2,1)];
732         elseif L1 <= L2
733                 AP_multi_values = [AP_multi_values; nan(L2-L1, l)];
734         end
735
736         AP_multi_values = [AP_multi_values, newAP];
737         AP_multi_legend = [AP_multi_legend; {strcat('Channel ', int2str(i_channel), ', AP ',
                int2str(i_AP))}];
738
739         plot(ha_AP_multi, AP_multi_values);
740         legend(ha_AP_multi, AP_multi_legend);
741             hold(ha_AP_multi, 'on');
742             plot(ha_AP_multi, zeros(size(AP_multi_values(:,1))), 'k');
743             hold(ha_AP_multi, 'off');
744         set(hfig_AP_multi, 'Visible', 'On');
745     end
746
747     function hbutton_AP_remove_Callback(~,~)
748         cla(ha_AP_multi);
749         if ~isempty(AP_multi_values)
750            [~, c] = size(AP_multi_values);
751            if c>1
752               AP_multi_values = AP_multi_values(:, 1:(end-1));
753               AP_multi_values = AP_multi_values(:,~all(isnan(AP_multi_values), 1));
754               AP_multi_legend = AP_multi_legend(1:(end-1),:);
755
756               plot(ha_AP_multi, AP_multi_values);
757               legend(ha_AP_multi, AP_multi_legend);
758                 hold(ha_AP_multi, 'on');
759                 plot(ha_AP_multi, zeros(size(AP_multi_values(:,1))), 'k');
760                 hold(ha_AP_multi, 'off');
761            else
762               AP_multi_values = [];
763               AP_multi_legend = [];
764               set(hfig_AP_multi, 'Visible', 'Off');
765            end
766         end
767     end
```

```
768
769     function hbutton_AP_quantif_Callback(~,~)
770         figures_AP(AP_data_struct);
771     end
772
773     function hbutton_AP_rasterplot_Callback(~,~)
774         figures_rasterplot(AP_data_struct);
775     end
776
777     function hcb_AP_Callback(~,~)
778         hedit_AP_Callback;
779     end
780
781
782 %% CALLBACKS VISUAL_BURST
783
784     function hbutton_burst_left_Callback(~, ~)
785         i_burst = max(1, i_burst-1);
786         set(hedit_burst, 'String', int2str(i_burst));
787         hedit_burst_Callback;
788     end
789
790     function hbutton_burst_right_Callback(~, ~)
791         i_burst = min(i_burst+1, bursts_count);
792         set(hedit_burst, 'String', int2str(i_burst));
793         hedit_burst_Callback;
794     end
795
796     function hbutton_burst_load_Callback(~, ~)
797         i_burst = 1;
798         cla(ha_burst);
799         set(hedit_burst, 'String', '1');
800         set(hbutton_burst_left, 'Visible', 'Off');
801         set(hbutton_burst_right, 'Visible', 'Off');
802         set(hedit_burst, 'Visible', 'Off');
803         set(hbutton_burst_go, 'Visible', 'Off');
804         set(hbutton_burst_go2, 'Visible', 'Off');
805         set(hbutton_burst_remove, 'Visible', 'Off');
806         set(hbutton_burst_quantif, 'Visible', 'Off');
807         set(hcb_burst, 'Visible', 'Off');
808         set(htext_file_data, 'Visible', 'Off');
809
810         set(htext_burst_data, 'Visible', 'Off');
811         set(htext_burst_start, 'Visible', 'Off');
812         set(htext_burst_length, 'Visible', 'Off');
813         set(htext_burst_max, 'Visible', 'Off');
814         set(htext_burst_AP_count, 'Visible', 'Off');
815
816         [burst_filename, burst_pathname, filterindex] = uigetfile({'*.m;*.mat', 'MATLAB␣
                Files␣(*.m,␣*.mat)'}, ...
817                                                                   'Pick␣a␣file␣containing␣
                                                                      EVENTS_burst');
818
819         if filterindex
820             burst_fullfilename = strcat(burst_pathname, burst_filename);
821             burst_data_struct = load(burst_fullfilename);
822             burst_data_struct = burst_data_struct.EVENTS_burst;
823
824             bursts_count = burst_data_struct.events_count;
825
```

```
826             set(htext_burst_file, 'String', burst_filename);
827             set(htext_file_data, 'Visible', 'On', 'String', strcat('Count of bursts : ',
                    int2str(bursts_count)));
828
829             if bursts_count
830                 set(htext_burst_file, 'String', burst_filename);
831                 set(hbutton_burst_left, 'Visible', 'On');
832                 set(hbutton_burst_right, 'Visible', 'On');
833                 set(hedit_burst, 'Visible', 'On');
834                 set(hbutton_burst_go, 'Visible', 'On');
835                 set(hbutton_burst_go2, 'Visible', 'On');
836                 set(hbutton_burst_remove, 'Visible', 'On');
837                 set(hbutton_burst_quantif, 'Visible', 'On');
838                 set(hcb_burst, 'Visible', 'On');
839                 set(htext_file_data, 'Visible', 'On', 'String', strcat('Count of bursts : ',
                        int2str(bursts_count)));
840
841                 set(htext_burst_data, 'Visible', 'On');
842                 set(htext_burst_start, 'Visible', 'On');
843                 set(htext_burst_length, 'Visible', 'On');
844                 set(htext_burst_max, 'Visible', 'On');
845                 set(htext_burst_AP_count, 'Visible', 'On');
846
847                 maximums = cellfun(@max, {burst_data_struct.cover}, 'UniformOutput', 0);
848                 burst_ymax = max(cell2mat(maximums));
849
850                 hedit_burst_Callback;
851             end
852         else
853             set(htext_burst_file, 'String', 'No loaded file...');
854         end
855     end
856
857
858     function hedit_burst_Callback(~, ~)
859         cla(ha_burst);
860
861         i_burst = min(max(str2num(get(hedit_burst, 'String')), 1), bursts_count);
862         burst_data = burst_data_struct(i_burst);
863
864         hold(ha_burst, 'on');
865         plot(ha_burst, burst_data.cover, 'b');
866
867         if get(hcb_burst, 'Value')
868             ymax = burst_ymax;
869         else
870             ymax = max(burst_data.cover);
871         end
872         for i = 1:120
873             plot(ha_burst, cell2mat(burst_data.values(i)) - burst_data.start, (i*ymax/120)*
                    ones(size(cell2mat(burst_data.values(i)))), ':mx');
874         end
875         set(ha_burst, 'Ylim', [0, ymax]);
876
877         set(hedit_burst, 'String', int2str(i_burst));
878
879         set(htext_burst_start, 'Visible', 'On', 'String', strcat('Start : ', int2str(
                burst_data.start), 'pts'));
880         set(htext_burst_length, 'Visible', 'On', 'String', strcat('Length : ', num2str(
                burst_data.length/25000*1000), ' ms'));
```

```
881         set(htext_burst_max, 'Visible', 'On', 'String', strcat('Amplitude : ', int2str(
                burst_data.peak), ' AP'));
882         set(htext_burst_AP_count, 'Visible', 'On', 'String', strcat('AP count : ', int2str(
                sum(cellfun(@length, burst_data.values))), ' AP'));
883     end
884
885     function hbutton_burst_go_Callback(~, ~)
886         newburst = cell2mat({burst_data.cover})';
887         [L1, l] = size(burst_multi_values);
888         L2 = length(newburst);
889
890         if L1 > L2
891                 newburst = [newburst; nan(L1-L2,1)];
892         elseif L1 <= L2
893                 burst_multi_values = [burst_multi_values; nan(L2-L1, l)];
894         end
895
896         burst_multi_values = [burst_multi_values, newburst];
897         burst_multi_legend = [burst_multi_legend; {strcat('Burst ', int2str(i_burst))}];
898
899         plot(ha_burst_multi, burst_multi_values);
900         legend(ha_burst_multi, burst_multi_legend);
901
902         set(hfig_burst_multi, 'Visible', 'On');
903     end
904
905     function hbutton_burst_go2_Callback(~, ~)
906         hbutton_burst_go_Callback;
907         randcolor = [rand rand rand];
908         for i = 1:120
909              plot(ha_burst_multi, cell2mat(burst_data.values(i)) - burst_data.start, (i*
                    burst_ymax/120)*ones(size(cell2mat(burst_data.values(i)))), ':mx', 'Color',
                    randcolor);
910         end
911     end
912
913
914     function hbutton_burst_remove_Callback(~,~)
915         cla(ha_burst_multi);
916         if ~isempty(burst_multi_values)
917            [~, c] = size(burst_multi_values);
918            if c>1
919               burst_multi_values = burst_multi_values(:, 1:(end-1));
920               burst_multi_values = burst_multi_values(:,~all(isnan(burst_multi_values), 1));
                       %ligne magique
921               burst_multi_legend = burst_multi_legend(1:(end-1),:);
922
923               plot(ha_burst_multi, burst_multi_values);
924               legend(ha_burst_multi, burst_multi_legend);
925            else
926               burst_multi_values = [];
927               burst_multi_legend = [];
928               set(hfig_burst_multi, 'Visible', 'Off');
929            end
930         end
931     end
932
933     function hbutton_burst_quantif_Callback(~,~)
934        figures_burst(burst_data_struct);
935     end
```

```
936
937         function hcb_burst_Callback(~,~)
938         hedit_burst_Callback;
939     end
940
941
942 %% CALLBACKS CAT
943
944     function hbutton_cat_load_Callback(~,~)
945
946         [filenames,cat_pathname,filterindex] = uigetfile({'*.m;*.mat', 'MATLAB Files (*.m, *.mat)'}, ...
947                                                          'Pick some files containing EVENTS_high', 'MultiSelect','On');
948
949         set(htext_cat_new, 'Visible', 'Off');
950
951         n_cat = 1;
952         M_cat = [0; 1];
953         time = 0;
954
955         if filterindex
956             if iscell(filenames)
957                 files_count = length(filenames);
958                 i_cat = 1;
959                 set(htext_cat_state, 'String', 'Concatenating...', 'Visible', 'On');
960                 set(htext_cat_load, 'String', filenames);
961                 set(hfig_cat_proc, 'Visible', 'On');
962                 set(htext_cat_step, 'String', strcat('Step ', num2str(i_cat), ' out of ', num2str(files_count)));
963                 set(htext_cat_file, 'String', filenames{i_cat});
964                 files = cell(files_count, 1);
965                 for j = 1:files_count
966                     files{j} = load(strcat(cat_pathname, filenames{j}));
967                 end
968             else
969                 set(htext_cat_load, 'String', 'Pick multiple files');
970             end
971         end
972     end
973
974     function hbutton_cat_cut_Callback(~,~)
975         n_cat = n_cat+1;
976         M_cat = [M_cat , [time + str2num(get(hedit_cat_left, 'String')); i_cat], [time + str2num(get(hedit_cat_right, 'String')); i_cat]];
977         set(hedit_cat_left, 'String','0');
978         set(hedit_cat_right, 'String', '0');
979         hbutton_cat_continue_Callback;
980     end
981
982     function hbutton_cat_continue_Callback(~,~)
983         time = time + files{i_cat}.recording_length;
984         if i_cat == files_count
985             M_cat = [M_cat, [time; files_count]];
986             cat_proc;
987         else
988             i_cat = i_cat+1;
989             set(htext_cat_file, 'String', filenames{i_cat});
990         end
```

```
991
992         set(htext_cat_step, 'String', strcat('Step ', num2str(i_cat), ' out of ', num2str(
                files_count)));
993     end
994
995     function cat_proc
996         retard = [0, files{1}.recording_length];
997         newfilenames = cell(n_cat, 1);
998
999         for i = 2:files_count
1000            retard = [retard, retard(end)+files{i}.recording_length];
1001        end
1002        for j = 1:n_cat
1003            filename = '';
1004            channels_count = 120;
1005
1006            EVENTS_high = [];
1007
1008            t1 = M_cat(1,2*j-1);
1009            t2 = M_cat(1,2*j);
1010            recording_length = t2-t1;
1011
1012            slave = struct('MEA', [], 'number_events', [0], 'start', [], 'length', [], '
                    values', [], 'min', [], 'peak_left', [], 'peak_right', [], 'diff_left', [],
                    'diff_right', [], 'halfway_length', [], 'ratio', []);
1013
1014            for l = 1:120
1015                EVENTS_high = [EVENTS_high; slave];
1016            end
1017
1018            for k = M_cat(2,2*j-1):M_cat(2,2*j)
1019
1020                filename = strcat(filename, ',', files{k}.filename);
1021
1022                delta = retard(k);
1023                for i = 1:120
1024                ind = find(and(files{k}.EVENTS_high(i).start+delta > t1, files{k}.
                        EVENTS_high(i).start+delta < t2));
1025                EVENTS_high(i).MEA = strcat(EVENTS_high(i).MEA, ',', files{k}.EVENTS_high(i)
                        .MEA);
1026                EVENTS_high(i).number_events = EVENTS_high(i).number_events + length(ind);
1027                EVENTS_high(i).start = [EVENTS_high(i).start, files{k}.EVENTS_high(i).start(
                        ind) - (M_cat(1,2*j-1) - retard(M_cat(2,2*j-1))) + (retard(k) - retard(
                        M_cat(2,2*j-1)))]; % attention au retard
1028                EVENTS_high(i).length = [EVENTS_high(i).length, files{k}.EVENTS_high(i).
                        length(ind)];
1029                EVENTS_high(i).values = [EVENTS_high(i).values, files{k}.EVENTS_high(i).
                        values(ind)];
1030                EVENTS_high(i).min = [EVENTS_high(i).min, files{k}.EVENTS_high(i).min(ind)];
1031                EVENTS_high(i).peak_left = [EVENTS_high(i).peak_left, files{k}.EVENTS_high(i
                        ).peak_left(ind)];
1032                EVENTS_high(i).peak_right = [EVENTS_high(i).peak_right, files{k}.EVENTS_high
                        (i).peak_right(ind)];
1033                EVENTS_high(i).diff_left = [EVENTS_high(i).diff_left, files{k}.EVENTS_high(i
                        ).diff_left(ind)];
1034                EVENTS_high(i).diff_right = [EVENTS_high(i).diff_right, files{k}.EVENTS_high
                        (i).diff_right(ind)];
1035                EVENTS_high(i).halfway_length = [EVENTS_high(i).halfway_length, files{k}.
                        EVENTS_high(i).halfway_length(ind)];
```

```
036                 EVENTS_high(i).ratio = [EVENTS_high(i).ratio, files{k}.EVENTS_high(i).ratio(
                        ind)];
037                 end
038             end
039
040
041             [newfilenames{j}, newpathname, filterindex] = uiputfile('*.mat', strcat('Save␣
                    the␣file␣', num2str(j), '␣as'), strcat(cat_pathname, 'PROCESSED_', filenames
                    {1}(1:(end-4)), '_', num2str(j), '.mat'));
042             if filterindex
043                 EVENTS_burst = burst_processing(filename, EVENTS_high, recording_length);
044                 save(strcat(newpathname, newfilenames{j}), 'filename', 'channels_count', '
                        recording_length', 'EVENTS_high', 'EVENTS_burst');
045             end
046         end
047         set(htext_cat_state, 'String', 'Concatenated␣!␣Load␣to␣concatenate␣again');
048         set(htext_cat_new, 'String', newfilenames, 'Visible', 'On');
049         set(hfig_cat_proc, 'Visible', 'Off');
050     end
051
052     function hbutton_cat_plot_Callback(~,~)
053         figures_rasterplot(files{i_cat});
054     end
055
056 end
```