Appendix 1

Data Browser Guide
## Table of Contents

I. **User Guide** .................................................................................................................. 42  
a) **Where downloading the GUI?** ...................................................................................... 43  
b) **How to launch the GUI?** .............................................................................................. 43  
c) **How to display traces?** ................................................................................................ 43  
   i. User tab .......................................................................................................................... 43  
   ii. Query tab ...................................................................................................................... 44  
   iii. Load tab ...................................................................................................................... 44  
e) **Browsing cells** .............................................................................................................. 45  
f) **Looking at one Trace** ................................................................................................... 46  
   i. Information concerning the Trace .................................................................................. 46  
   ii. Features ....................................................................................................................... 46  
II. **Technical Guide** ......................................................................................................... 47  
a) **How are sorted the matlab files?** .............................................................................. 47  
b) **How come the data?** .................................................................................................. 47  
   i. TraceBD connection and Query .................................................................................... 48  
   ii. SRB .............................................................................................................................. 48  
   iii. WaveReader .............................................................................................................. 48  
c) **How is done the features extraction** ......................................................................... 49  
   i. C file ............................................................................................................................. 49  
   ii. Mex file ........................................................................................................................ 49  
   iii. FigureGUI.m ............................................................................................................ 49  
A. **Matlab functions** ....................................................................................................... 50
### Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>Start Page</td>
<td>43</td>
</tr>
<tr>
<td>37</td>
<td>User Tab</td>
<td>43</td>
</tr>
<tr>
<td>38</td>
<td>Query Tab</td>
<td>44</td>
</tr>
<tr>
<td>39</td>
<td>Load Tab</td>
<td>45</td>
</tr>
<tr>
<td>40</td>
<td>Options Tab</td>
<td>45</td>
</tr>
<tr>
<td>41</td>
<td>Main Page with Traces</td>
<td>45</td>
</tr>
<tr>
<td>42</td>
<td>One trace with features</td>
<td>46</td>
</tr>
<tr>
<td>43</td>
<td>Relation between the main matlab functions</td>
<td>47</td>
</tr>
<tr>
<td>44</td>
<td>Matlab, TraceDB and SRB connections</td>
<td>48</td>
</tr>
</tbody>
</table>
I. User Guide

a) Where downloading the GUI?

The GUI data browser files correspond to a directory containing mainly matlab files, and just few C files. This package is available on SVN in:
analysis/trunk/DataBrowser/TraceGUI/
The user needs to download the entire directory “TraceGUI”.

b) How to launch the GUI?

The directory contains a matlab file called “testGUI.m”. The user just need to launch it on matlab, then a window will appear with the different options to select.

c) How to display traces?

Once the window is opened the user has to select some parameters in order to display the traces he wants. These parameters are sorted in different tabs

i. User tab

First the user has to enter his user name and his password to connect to the database. It will allow him to download traces and also to save features in the database.
ii. **Query tab**

This tab presents 7 popup menus or textbox which allow the user to choose the traces he wants to look at.

- The first popup menu proposes different names of experimentalists, it corresponds to the ones whose the experiences have been stored in the database. On the actual version, Maria and Claudia works well, concerning the others some problems appear because of WaveNotes missing or different cell’s names. One experimentalist’s name is “simulation”, it allows the user to load traces from simulated cells, in this case the user has to choose the date from the simulation. The date has to written like that: 12.02.07. The fact to choose an experimentalist is compulsory to display a trace, if the user doesn't the GUI won't work.

- The second popup menu proposes to the user to choose within the 21 different eCodes (one eCode corresponds to a given stimulus, for more information look at the eCode specification). All the eCodes are not actually available for the experimental cells, but concerning the simulated cells the different eCodes can be displayed. It is compulsory to choose an eCode to display a trace, if the not the GUI won't work.

- The third popup menu proposes to the user to choose within the different eTypes available. The eType corresponds to the electro physiological behavior of the cell. They are classified according to the Petilla classification. The eTypes proposed on the GUI are directly extracted from the database. The choice of the eType is not compulsory to display a trace.

- The next popup menu allows the user to choose within all the morphological types present in the database. The list of the mType is automatically extracted from Neobase. The choice of the mType is not compulsory to display a trace.

- The next popup menu proposes to the user to choose the layer from the cell he wants to display the traces. The names of the layer are extracted from the database, but there is a problem of normalization. The layer 5 works well. It is not compulsory to choose a layer.

- The user can directly choose the cell he wants to look at. For this he just has to select an experimentalist, an eCode and then enter the name of the cell. The name should be like that: C010305A1. If no name has been enter, the data browser will take the first cell it finds according to the given criteria.

- The user can choose to work on his local directories or to connect to SRB, then he just have to select the server on which he wants to work. The default server is SRB.

![Figure 38 Query Tab](image)

iii. **Load tab**

Once all the parameters have been chosen the user can click on “Load Traces”. Then the GUI will connect to SRB, download the traces and will display them in the window. The second button “Load stimulus” can be used once the traces has been displayed, it allows the user to open in a new figure the current traces.
iv. **Options tab (fig 1.5)**

Not useful to display the traces. It just allows the user whether he wants, to save the traces he is looking at. The traces are saved as an ascii file or a mat file.

---

d) **What is displayed?**

Once the user click on “Load traces”, the traces will be displayed the central part of the window (fig 1.6). Generally for one eCode there is more than one trace. It often appears that the experimentalist did few series of experiences with for each different steps of stimulus. If we take the example of IDrest, we can have 25 traces, this corresponds to five series of experiences with for each 5 different stimuli. So the GUI will display a 5x5 matrix of traces where the lines correspond to the series and the columns to the steps of stimulus.

---

e) **Browsing cells**

On the bottom on the window, two command buttons are present. They allow the user to display the next cell or to display the old one. If the user can't anymore go back or next, a message box appears to say that there is no more trace.

The user can also go through the different eCodes of one cell.
f) **Looking at one Trace**

The user can display a trace in a new window by just clicking on it. Then he can look more precisely at it and extract some features.

i. **Information concerning the Trace**

On the left part of the new window, the user can see a column with some information. This information corresponds to the WaveNotes. These notes have been filled by the experimentalist during the experience. It can indicate for example the date of the trace, the eCode, the stimulus applied to the cell, the size of the trace vector... It could be useful to understand better the trace, the WaveNotes are also used for the features extracting.

ii. **Features**

In the upper left corner the user has the possibility to extract some features from the trace. The extraction functions are already defined in a C program (for more information see the technical part). They correspond to the parameters of the APs (start, peak, end, after hyper polarization, rise time, fall time ...) but also to the frequency and to some mean values.

Once the user clicks on the button “Extract Features” some of the features appear on the right part, on the tab pages 1, 2 and 3. Then the user can click on “AP points” to display on the trace the start, peak, end and ahp points, he can also click on “V_mean / V_rest” to display the mean of the Voltage before the stimulus and the one during the stimulus. And to finish concerning the features, the user can click on “ISI histogram” to display a new figure with the evolution of the ISI along the trace. But the user needs to click first on Extract Features to display the other parameters.

On the upper right corner, the user can click on “load Current” to display in a new window the current stimulus applied to the cell.

---

Figure 42 One trace with features
II. Technical Guide

This part concerns more the programming aspect of the data browser. This means how are sorted the different files, how they connect each other, how does it download the trace data and also how is build the extract feature function.

a) How are sorted the matlab files?

The data browser directory contains some files and four directories: C_features, eCodeBrowser, FeatureSave and lib. The first one contains the C files used for the feature extraction, the second one concerns the .m files used to display trace from simulated cells, the third one will contains the features calculated from the traces and the last is a library used to convert the .ibw trace in vector readable by matlab.

Concerning the .m files and how they are connected together, the next schema (fig 2.1) gives a global idea of how the main functions are related:

Figure 43 Relation between the main matlab functions

b) How come the data?

The figure 2.2 shows how the matlab program, the database and SRB (where are stored the traces) are linked together.
Figure 44 Matlab, TraceDB and SRB connections

i. **TraceBD connection and Query**

The GUI connects to Neobase where are stored traces from Maria, brandi, Claudia... This connection is done thanks to a mex file done for matlab and called “mym”. It allows to connect to bbpteam.epfl.ch by giving a login and a password, and then to access to Neobase database. Furthermore this function takes MySQL queries as input. A Matlab function takes as input the parameters entered by the user and returns the appropriate query. Then the mym function can be applied, it will return a vector with the path to SRB of all the traces corresponding to the query of the user.

ii. **SRB**

Once the program has the path to SRB it needs to connect to it, for this it uses a java object created by Laurent Francioli called SRBConnector which can access to SRB. It will then download the trace in Igor Binary Wave format which is not readable on matlab. So the program needs a new function.

iii. **WaveReader**

WaveReader is a java file created by Laurent Francioli which is able to transform an igor binary wave file into a vector readable on matlab. The waves are converted in a wave java object on which it can be applied different methods to extract all the informations contained in the binary file. So it can extract the trace points but also the units, the sampling, the start time, the end time and the waveNotes.
c) **How is done the features extraction**

In order to have a program working on different supports, the features extraction has been implemented on C and not on matlab.

i. **C file**

The extraction functions are divided into three files. The first one is called matlab.c, it provides functions like max or mean already implemented on matlab but not existing on C. The second one is called Primary.c, it contains all the functions used to calculate the primary features like the parameters of the start point, the end point... The primary features are features calculated directly from the trace and not from intermediate parameters. The last file is called Secondary.c, it contains functions used to calculate the secondary features like ISI, frequency... The secondary features are features determined from the primary features. Then all these functions needs to be implemented together in order to extract the features from the traces.

ii. **Mex file**

The mex file, is a C file which allows to import C functions on matlab. This mex file is called extractFeatures.c, it applies all the primary and secondary features functions and then creates a function called extractFeatures which can be applied from matlab. This function takes 6 inputs which are the voltage vector and his length, the start time and the end time of the stimulus, the dt and the current vector. And it returns 3 outputs which are an array of AP structures, a vector with the values of the Inter Spikes Intervals and a vector with general features. The AP structure contains some fields with in each the value of a given parameter like the start time, the ahp time...

iii. **FigureGUI.m**

Once the features are extracted, they are displayed on the GUI which contains just the trace. It automatically open the Features structure and go through the different fields, for each it creates a panel and a text box to display it. Then if another user wants to add his own extractFeature function, if this one has the same global construction, the GUI will be able to display these new features.
A. Matlab functions

% Name : autoExtractInputRes.m
% Author : Aurelien Mace
% Date : June 2007

% Goal :
% This script is used to extract the input resistance form the IV curve. It
% can easily be transformed as a function taking in input the name of the
% cell
% Implementation :
% it first connect to the database thanks to the function mym, then it
% query it and get the name fo the files wanted.
% The function getTrace is used to get the Voltage and Current traces with
% the WaveNotes. Contrary to the other mex function the extractFeaturesIV
% takes all the voltage and current traces as input. In fact the resistance
% as to be calculated on all of them

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name : autoExtractInputTau.m
% Author : Aurelien Mace
% Date : June 2007

% Goal : Return for each eType within each mType the mean value of the time
% membrane constant from all the cells classified with a certainty of
% 100%

% Implementation : the script goes through all the etype and mtype present
% in the database then for each combination it takes all the cells
% and extract from them the time membrane constant. Then it does the mean
% and put in a structure.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name : autoFeatClassi.m
% Author : Aurelien Mace
% Date : July 2007

% Goal : Extract features for all the cells of the database classified with
% a certainty fixed by the variable of the same name. It save the features
% in 2 files one for features and one for the ISI.
% Implementation : It connects first to the database. Then goes through all
% the etype and mtype and for each combination query the cell names. Then
% it extracts the traces points and applied the extraction features
% function. It finally stores it in structure which then saved as a .mat
% file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name : autoFillDB.m
% Author : Aurelien Mace
% Date : June 2007
%
% Goal : Automatically fill the database with the features data
% Implementation : It connects to the database. Then it goes through all
% the etypes and mtypes. It goes through all the cells of the database and
% extract the features from them to finally use the function fillDataBase
% to put the data in Neobase.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name : autoExtractFeatures.m
% Author : Aurelien Mace
% Date : May 2007
%
% Goal : Automatically features extraction from the traces and save them as .mat files or
% or .asc file. The file are save in the directory /FeatureSave. This
% function was created before Neobase at this time it was not possible to
% store the feature in the database
%
% Implementation : It connects to the database. Then takes the cell names
% form a file called testCell.txt. It converts the list of the names and eTypes into a
% cell format. Then it goes through all the cells and extract features form
% them to store them in asc or mat files

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name : automaticQualityCheck.m
% Author : Aurelien Mace
% Date : June 2007
% Goal: Check the quality of the trace contained in Neobase and plot a
% graph with the different criteria and the percentage of bad cells for
% each. The graph produced is used for the calibration plan

% Implementation: It connects to the database. Goes through all the eType
% and mTypes, Applies the Quality Check function and then save the result
% in a .mat file. Finally it uses this .mat file to plot a bar graph with
% the percentage of bad cells ofr each criteria

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name: autoStat.m
% Author: Aurelien Mace
% Date: June 2007

% Goal: Extract the mean and standard deviation for the fitness features
% and the generalization features for each etype within each mtype. This
% stat are saved in the .mat file called BioStat.

% Implementation: It query the database goes through all the mtype and
% etype, extract the features, selects the fitness and generalization one,
% calculate for each etype the mean and the standard deviation and then
% store these values in a structure saved as a .mat file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name: autoStatCal.m
% Author: Aurelien Mace
% Date: June 2007

% Goal: Plot the percentage of error between Model cell and biology cells.
% It also plot a table with the features presenting the most percentage of
% error. Used for the calibration plan

% Implementation: It opens to .amt file wich contain the stat on the model
% and bio cells. Then it calculates the percentage of error and plot it.
% Name : autoStatCal.m
% Author : Aurelien Mace
% Date : June 2007

% Goal : Plot statistics on the database, like the number of cells within
% each etype and mtype.

% Implementation : It query the database, count the number of cells for
% each etype within each mtype and plot pie and table.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name : autoStatSim.m
% Author : Aurelien Mace
% Date : June 2007

% Goal : Extract the mean and standard deviation for the fitness features
% and the generalization features for each etype within each mtype from the model cells. This
% stat are saved in the .mat file called SimStat.

% Implementation : It goes to the directory where the wanted traces are
% stored, then it extract the traces points, extract the features and then
% calculated the mean and standard deviation for the fitness features and
% the generalization features

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name : cutWaveNote.m
% Author : Aurelien Mace
% Date : April 2007

% Goal : Transform the string wave notes of the meta data into a cell
% format where each cell of the first column corresponds to a parameter and
% each cell of the 2nd column to the corresponding value.

% Implementation : Takes as input the wave note string, count the number of
% ; and # to see if it is in Pulse Q format or not. Then it parse the note
% with the ; or # delimiter. It returns a cell with the wave note parameter
% and value

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name : dataBasemTeT.m
% Author : Aurelien Mace
% Date : June 2007

% Goal : Plot the quantity of cells for each etype within each mtype for
% Maria cells. Thos plot is then used in the Calibration plan
% Implementation : It connects to the database, select the etype and mtype
% where there are cells. Goes through all of them, then compt the number
% of cells, and finally plot a table where the line corresponds to the
% mtype and the columns to the etype and where each square has a color
% giving the number of cells. Then the plot is saved as a png file.

% Name : dataBasemTeT.m
% Author : Aurelien Mace
% Date : May 2007

% Goal : Stored in a ascii file the name of the cells in Neobase with there
% mtype and etype classification and for each with the percentage of
% certainty
% Implementation : It connects to neobase, create an ascii file, goes
% through all the etype and mtype, query the database for the cell name
% and certainty and then fill the ascii file

% Name : extractPulse.m
% Author : Aurelien Mace
% Date : April 2007

% Goal : Extract the pulse parameter from the pulse definiton. The pulse
% parameters wanted are the time satrt and time end
% Implementation : Read the pulse definition, parse it and extract the two
% times wanted.
% Name: extractRamp.m
% Author: Aurelien Mace
% Date: April 2007

% Goal: Extract the ramp parameter from the ramp definition. The ramp
% parameters wanted are the time start, the time end, the start amplitude
% and the end amplitude

% Implementation: Read the ramp definition, parse it and extract the
% parameters

% Name: extractSegment.m
% Author: Aurelien Mace
% Date: April 2007

% Goal: Extract the segment parameter from the segment definition. The segment
% parameters wanted are the time start, the time end and the amplitude

% Implementation: Read the segment definition, parse it and extract
% parameters

% Name: FigureGUI.m
% Author: Aurelien Mace
% Date: June 2007

% Goal: Display a window where the user can look at one trace, extract
% features from it and then displays them.

% Implementation: I use the matlab GUI tools, which can be open thank to
% the command 'guide'. Then I implemented the function of each button.

% Name: fillDataBase.m
% Author: Aurelien Mace
% Date: June 2007
Goal: Function used within the script autoFillDB. The goal of this function is to fill the table TraceActionPotential, TraceISIVector and TraceFeatures for one trace of one cell.

Implementation: This function takes as input the name of the trace, the name of the cell, the AP points, the ISI vector and the feature structure. Then it query neobase to get the id of the trace and finally fill the different tables.

Goal: Find the value in the wave notes of the given parameter

Implementation: Takes as entry the WaveNote as a cell format and the name of the parameter. Then it search in the wave note the index of the parameter and return the corresponding value

Goal: Return the number of the channel for the given path to a trace

Implementation: Takes as input the path, and parse it to find the number of the channel

Goal: return the position of the cursor in the main window (the one with all the traces). The position of the cursor corresponds to the index of the trace on which the user has click. This position is then used to
% open the good trace in a new window

% Implementation : Used the matlab function current point which the x and y
% of the cursor. Then the widow in divided as a table with the given
% number of line and column (input value). Then the index of the curso is
% found

% Name : getNameCell.m
% Author : Aurelien Mace
% Date : April 2007

% Goal : give the name of the cell for one trace given his path to srb
% Implementation : Takes as input the path and parse it to get the cell
% name

% Name : getnumTrace.m
% Author : Aurelien Mace
% Date : March 2007

% Goal : give the number of the trace written in the path
% Implementation : Takes as input the path and parse it to get the number
% of the trace

% Name : getQuery.m
% Author : Aurelien Mace
% Date : July 2007

% Goal : returns the Query corresponding to the given parameters
% Implementation : Takes as input parameter of the query and in function on
% it write a SQL query
% Name : getStartEnd.m
% Author : Aurelien Mace
% Date : April 2007

% Goal : returns the start and end time of the stimulus

% Implementation : Takes as input the Wave notes and the dt. Then if
% unction of the type of stimulus (pluse, ramp or segment) it applies the
% extractPulse ... functions to get the wanted value

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name : getTime.m
% Author : Aurelien Mace
% Date : April 2007

% Goal : Gives the hour of the experiment by looking at the wavenotes

% Implementation : parse the wave note to get the hour of the experiment

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name : getTraces.m
% Author : Aurelien Mace
% Date : April 2007

% Goal : Return the traces and wavenotes for one cell, given the list of trace and
% the index of the first trace.

% Implementation : Takes as input list of the trace file, the index of the
% first trace in this list and the variable local which corresponds to the fact if
% the user works on a server or on a local directory. First the function
% connects to srb then it goes through all the trace name until the name of
% the cell changes or until the end of the list of traces. For each trace
% file it applies the function tracePlotGui which return the traces points
% and the wave notes. Then it stores it in cells, one for the voltage
% traces and one for the current traces, the wave note and name, X and Y
% units and the number of the traces are also stored.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name : LaunchBackFigure.m
% Author : Aurelien Mace
% Date: May 2007

% Goal: Returns the current, voltage, wave notes information of the cell before the displayed one in the database. This function is used in the GUI testGUI.m for the button back.

% Implementation: Takes the list of traces corresponding to the given query. Takes the names of the current cell and search in the list of trace the name of the one before, once it has found it extract the traces and the wave notes

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name: LaunchFigure.m
% Author: Aurelien Mace
% Date: June 2007

% Goal: Connect to srb and applies the function getTraces to get the traces points for the currents and voltage and also the wave notes

% Implementation:

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name: LaunchGUI.m
% Author: Aurelien Mace
% Date: June 2007

% Goal: Add the good java path and connects to the database

% Implementation:

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name: LaunchNextFigure.m
% Author: Aurelien Mace
% Date: May 2007

% Goal: Returns the current, voltage, wave notes information of the cell after the displayed one in the database. This function is used in the GUI testGUI.m for the button next.
% Implementation: Takes the list of traces corresponding to the given
% query. Takes the names of the current cell and search in the list of
% trace the name of the one after, once it has found it extract the traces
% and the wave notes

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name: LaunchTrace.m
% Author: Aurelien Mace
% Date: July 2007

% Goal: Draw the traces of the wanted cell on the GUI

% Implementation: It first extracts the information of the wave notes to
% know the parameters of the stimulus (the number of step and the number of
% repetitions), this information will be useful to place the traces on
% the window. Each line correspond to a repetition of the stimulus and each
% column corresponds to a step of the stimulus. Then the good traces are
% plot at the good place.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name: myfunctionGUI.m
% Author: Aurelien Mace
% Date: May 2007

% Goal: Allows the user to open a trace in a new special window which allows him
% to extract features. This function is applied when the user click in the
% main GUI

% Implementation: Takes as input the parameter of the main GUI, like all
% the traces, the number of line and columns, the list of the wave note and
% the GUI object (the variable a and s are not used). It takes in first the
% place of the cursor in the window, then in function of this position it
% opens the good trace in a new GUI called figureGUI. This new GUI will
% display the trace and the wave notes, it also allows the user to extract
% features

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name: myfunctionGUIMSim.m
% Author: Aurelien Mace
% Date: May 2007

% Goal: Allows the user to open a trace in a new special window which allows him to extract features. This function is applied when the user click in the main GUI

% Implementation: Takes as input the parameter of the main GUI, like all the traces, the number of line and columns, the list of the wave note and the GUI object (the variable a and s are not used). It takes in first the place of the cursor in the window, then in function of this position it opens the good trace in a new GUI called figureGUI, this new GUI will display the trace and the wave notes, it also allows the user to extract features.

% Name: normalizR.m
% Author: Aurelien Mace
% Date: July 2007

% Goal: Normalize the rows of a matrix between 0 and 1

% Implementation: Takes as input the matrix. It extract the min and max value of each row, substract the min value and divides by max-min

% Name: QualityCheckOneTrace.m
% Author: Aurelien Mace
% Date: July 2007

% Goal: Check the quality of one trace by looking at his features. For more information concerning Quality check control, please have a look at the documentation 'Quality check'

% Implementation: Takes as input the AP points, the ISI, the Features structure, the end time of the stimulus and the voltage. Then it checks each parameters and return a vector containing the result for all the parameters, the value 1 means OK the value 0 means bad.
% Name : QualityCheckV2.m
% Author : Aurelien Mace
% Date : July 2007

% Goal : Checks the quality of one cell by looking at his features. Used in
% the script automaticQualityCheck.m. For more information concerning
% Quality check control, please have a look at the documentation 'Quality
% check'

% Implementation : Takes as input the paths to the traces of one cell
% Then it extracts the AP points, the ISI, the Features
% structure. Then it checks each parameters and return a vector containing the result for all the
% parameters, the value 1 means OK the value 0 means bad. finally it does
% the mean of all the traces

% Name : statFeatures.m
% Author : Aurelien Mace
% Date : July 2007

% Goal : Do stat on the features extracted form given etype and mtype. It
% plot cloud points for combination of two parameters in order to see if
% there could be some cluster in the etype class

% Implementation : Open the file FeatExc20072.mat created thank to the
% script autoFeatClassi.m. Then it takes from this file the fitness and
% generalization features. It normalizes it and display each one against
% each of the different etype classes. It give cloud points with different
% color, each color correspond to an etype.

% Name : testGUI.m
% Author : Aurelien Mace
% Date : July 2007

% Goal : GUI whcih allows the user to query cell and look at their traces
% Implementation: it uses the GUI toolbox from MATLAB, this toolbox can be
% launched with the command 'guide'

% Name: tracePlotGUI.m
% Author: Aurelien Mace
% Date: June 2007

% Goal: Return a vector of values for the trace but also the waveNote and the
% Units given the path to srb or to a local directory

% Implementation: It extracts the binary file from srb, used java
% function to convert it in MATLAB format and extract from this file the
% data points, the units, the wave notes.

% Name: getCursorPosition.m
% Author: Aurelien Mace
% Date: March 2007

% Goal: return the position of the cursor in the main window (the one
% with all the traces). The position of the cursor corresponds to the index
% of the trace on which the user has clicked. This position is then used to
% open the good trace in a new window

% Implementation: Used the MATLAB function current point which the x and y
% of the cursor. Then the window is divided as a table with the given
% number of line and column (input value). Then the index of the cursor is
% found

% Name: getfileData.m
% Author: Aurelien Mace
% Date: March 2007
% Goal : Extract the information contained in the hdf5 file of the
% simulated trace

% Implementation : Use the hdf5read function of matlab

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name : geth5.m
% Author : Aurelien Mace
% Date : March 2007

% Goal : get the hdf5 file cell name from the list of files in the directory where
% the simulated cells are stored

% Implementation : parse the list of files

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name : getMeType.m
% Author : Aurelien Mace
% Date : March 2007

% Goal : Return the metype of the cell wanted

% Implementation : read the MeType.txt file where all the metype of all the
% simulated cells are stored. It takes as input the names of the file where
% are the metype, the name of the cell nad the path to the directory where
% is stord the file.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name : getStimulus.m
% Author : Aurelien Mace
% Date : March 2007

% Goal : Return the stimulus parameter

% Implementation : read the MeType.txt file where all the stimulus
% are stored. It takes as input the names of the file where
% are the stimulus, the name of the trace and the path to the directory where
%% is stored the file.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name : getTrace.m
% Author : Aurelien Mace
% Date : March 2007

% Goal : returns the trace point and the trace info like the metype, the
% stimulus...

% Implementation : Takes as input the list of the list of the trace, the
% name of the cell and the path to the directory where are stored the
% traces. Then it used the function getfileData to get the points and some
% other information, then it use the function getMeType to get the metype
% and finally the function getStimulus to get the stimulus parameters

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name : launchBackFigureSim.m
% Author : Aurelien Mace
% Date : March 2007

% Goal : plot the simulated cells before the ones currently plotted, in
% function of the parameters chosen by the user.

% Implementation : Takes the parameters chosen by the user, go to the
% directory where are stored the cells, takes the cell before the one
% plotted according to the parameters and then plot the traces.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Name : launchFigureSim.m
% Author : Aurelien Mace
% Date : March 2007

% Goal : plot the simulated cells according the parameters chosen by the user.

% Implementation : Takes the parameters chosen by the user, go to the
% directory where are stored the cells, takes the first cell corresponding
% to the parameters and then plot the traces.
% Name : launchNextFigureSim.m
% Author : Aurelien Mace
% Date : March 2007

% Goal : plot the simulated cells after the ones currently plotted, in
% function of the parameters chosen by the user.

% Implementation : Takes the parameters chosen by the user, go to the
% directory where are stored the cells, takes the cell after the one
% plotted according to the parameters and then plot the traces.

% Name : myfunctionGUISim.m
% Author : Aurelien Mace
% Date : May 2007

% Goal : Allows the user to open a trace in a new special window which allows him
% to extract features. This function is applied when the user click in the
% main GUI

% Implementation : Takes as input the parameter of the main GUI, like all
% the traces, the number of line and columns, the list of teh wave note and
% the GUI object (the variable a and s are not used). It takes in first the
% place of the cursor in the window, then in function of this position it
% opens the good trace in a new GUI called figureGUI, this new GUI will
% display the trace and the wave notes, it also allows the user to extract
% features
Appendix 2

GENERAL APPROACH TO FEATURE EXTRACTION
Table of contents

I. Rationale & Approach ................................................................. 70
   A. Rationale: ........................................................................... 70
   B. Approach: .......................................................................... 70
   C. List of all features .............................................................. 70
II. PA Features ............................................................................ 72
    A. PA Features Currently implemented .................................. 72
III. PL FEATURES ........................................................................ 83
    A. PL Features currently implemented ................................... 83
IV. SA FEATURES ........................................................................ 86
    A. SA features currently implemented ................................... 86
    B. SA features under consideration ...................................... 107
V. SL FEATURES .......................................................................... 108
    A. SL FEATURES Currently implemented .............................. 108
VI. APPLICATION ........................................................................ 109
I. Rationale & Approach

A. Rationale:
The aim of this part is to see how feature extraction would be needed in biology and simulation experiments. From analyzing these features, one can try to find a good definition of types of neurons, i.e., by analyzing the time of each spike relative to onset of current, one can try to find a good definition of bursting vs nonbursting, adapting vs nonadapting based on secondary statistics, such as early to late ISI ratios.

B. Approach:
Those fields which starts with P# are the primary features obtained directly from the voltage and current traces and those with S#, the secondary, obtained from the primary traces. All of these features could be used in the future to discriminate between different types of cells. Moreover, both, primary and secondary features are divided into two groups:

a) PA# Feature: Primary feature, potentially extractable from ALL voltage recordings
b) PL# Feature: Primary features with meaning LIMITED to the context of special experimental protocols.
c) SA# Feature: Secondary features, potentially extractable from ALL recordings
d) SL# features: Secondary features, extractable from LIMITED PL features

C. List of all features
For each cell, a matlab structure will be created with the following features:

(For more informations concerning the structure, please look at the document called “Extract Features Structure”)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA01 pk_timing</td>
<td>% Time of each spike</td>
</tr>
<tr>
<td>PA02 ap_pk</td>
<td>% AP height</td>
</tr>
<tr>
<td>PA03 width</td>
<td>% AP width at the half of each spike</td>
</tr>
<tr>
<td>PA04 start_timing</td>
<td>% Time of the start of each AP</td>
</tr>
<tr>
<td>PA05 ap_start</td>
<td>% Voltage value of the start point of each AP</td>
</tr>
<tr>
<td>PA06 end_timing</td>
<td>% Time of the end of each AP</td>
</tr>
<tr>
<td>PA07 ap_end</td>
<td>% Voltage value of the end point of each AP</td>
</tr>
<tr>
<td>PA08 ahp_depth</td>
<td>% ahp depth value when there is an ahp</td>
</tr>
<tr>
<td>PA09 ahp_time</td>
<td>% Time of each ahp</td>
</tr>
<tr>
<td>PA10 v_mean</td>
<td>% Mean value of the voltage during the stimulus</td>
</tr>
<tr>
<td>PA11 v_rest</td>
<td>% Mean value of the voltage before the stimulus (resting potential)</td>
</tr>
<tr>
<td>PL01 accRest</td>
<td>% Access Resistance, just fro IDrest</td>
</tr>
<tr>
<td>PL02 membrane_time</td>
<td>% Time membrane constant for IV</td>
</tr>
<tr>
<td>PL03 in_res</td>
<td>% Input resistance</td>
</tr>
<tr>
<td>SL01 first_sp_delay</td>
<td>% Delay to first spike (relative to the start of the current).</td>
</tr>
<tr>
<td>SA01 rise_time</td>
<td>% Time of between the start and the peak</td>
</tr>
<tr>
<td>SA02 fall_time</td>
<td>% Time of between the peak and the end</td>
</tr>
<tr>
<td>SA03 rise_slope</td>
<td>% slope of the rise part</td>
</tr>
<tr>
<td>SA04 fall_slope</td>
<td>% slope of the fall part</td>
</tr>
<tr>
<td>SA05 ahp_delay</td>
<td>% Time between the end of the AP and the ahp</td>
</tr>
<tr>
<td>SA06 ahp_slope</td>
<td>% slope between the ahp and the next start</td>
</tr>
<tr>
<td>SA07 ISI</td>
<td>% Inter Spikes Interval values</td>
</tr>
<tr>
<td>SA09 isi_initial_to_middle</td>
<td>% rate between first AP isi and middle AP isi</td>
</tr>
<tr>
<td>SA10 isi_middle_to_final</td>
<td>% rate between middle APs isi and last APs isi</td>
</tr>
<tr>
<td>SA11 isi_percentage</td>
<td>% Percentage of variation of the ISI between beginning and</td>
</tr>
</tbody>
</table>

70
SA12 : frequency [float] % Frequency of the APs
SA13 : ap_drop [float] % Drop between the first and the second AP
SA14 : ap_width_mean [float] % Mean of the width of the APs
SA15 : ap_height_mean [float] % Mean of the AP height values
SA16 : rise_time_mean [float] % Mean of the rise time
SA17 : fall_time_mean [float] % Mean of the fall time
SA18 : rise_slope_mean [float] % Mean of the rise slope
SA19 : fall_slope_mean [float] % Mean of the fall slope
SA20 : ahp_delay_mean [float] % Mean of the ahp_delay
SA21 : ahp_depths_mean [float] % Mean of the ahp_depths_mean
SA22 : ahp_slope_mean [float] % Mean of the ahp slope
SA23 : adap_ind [float] % The adapting index
SA24 : ap_grad [float] % The adapting indexGradient of the peaks values (has to be implemented)
II. PA Features

A. PA Features Currently implemented

<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA01</td>
<td>Peak Time</td>
<td>[1xN double]</td>
<td>Primary Feature. Array of the time of the peaks of each spikes.</td>
<td>scripts</td>
</tr>
</tbody>
</table>

Rationale & Comments:
A spike is detected thanks to the derivative. To be detected a spike should have a derivative over 20% of the max derivative value, and a height spike over -20mV in order to avoid small artifact.

![Figure pk_timing](Too long)

**PSEUDOCODE**
Calculate Derivative

```
maxD = max(Derivative)
position = FIND(Derivative > maxD*20%)
for i=0;i<size(Position)-1;i++
    if position(i+1)-position(i)>1
        for j=position(i);j<position(i+1);j++
            FIND time of max Voltage
        end
    end
end
```
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA02</td>
<td>Peak height</td>
<td>[1xN float]</td>
<td>Primary Feature. Array of the height value of each spikes</td>
<td></td>
</tr>
</tbody>
</table>

**Rationale & Comments:** Figure ap_pk:

**PSEUDO CODE**

\[
\text{pk\_height} = \text{Voltage(pk\_timing)}
\]

**EXAMPLE CODE**

```c
void AP_peaks_heights(float * AP_pk, float *Voltage, int AP_leng, double * AP_pk_timing)
{
  int i;
  int index;
  for (i=0; i<AP_leng; i++)
  {
    index = (int)AP_pk_timing[i];
    AP_pk[i] = Voltage[index];
  }
}
```
Short Name | Full Name | Format | Description | Related E-code scripts
--- | --- | --- | --- | ---
PA03 | Width at half amplitude | [1xN double] | Primary Feature. Array of the width of each AP at the half amplitude | Figure width:

Rationale & Comments:
In order to really take the half amplitude we do an extrapolation around the half value. In fact the sampling is so small that there is no experimental point for the half amplitude.

PSEUDOCODE
half_amp = ap_start + (ap_pk - ap_start)/2
Extrapolation for half_amp_rise -> time_half_rise
Extrapolation for half_amp_fall -> time_half_fall
width = time_half_fall - time_half_rise

EXAMPLE CODE
```c
void half_width(double * half_width, float * AP_start, double* AP_start_timing, float * AP_pk, int AP_leng, float * Voltage, int volt_leng, double dt) {
    int i,start;
    double half_value;
    double deriv;
    double half_time1, half_time2, V1, V2;
    for(i=0; i<AP_leng; i++) {
        half_value = AP_start[i] + (AP_pk[i] - AP_start[i])/2;
        start = AP_start_timing[i];
        while(Voltage[start]<half_value) {
            start++;
        }
        V1 = Voltage[start-1];
        V2 = Voltage[start];
        deriv = (V2-V1)/dt;
        half_time1 = (start-1)*dt + (half_value - Voltage[start-1])/deriv;
        while(Voltage[start]>half_value) {
            start++;
        }
        V1 = Voltage[start];
        V2 = Voltage[start-1];
        deriv = (V2-V1)/dt;
        half_time2 = (start)*dt - (half_value - Voltage[start])/deriv;
        half_width[i] = half_time2 - half_time1;
    }
}
```
Short Name | Full Name | Format | Description | Related E-code
---|---|---|---|---
PA04 | Start time | [1xN double] | Primary Feature. Array of the time of the peaks of each spikes. | scripts

Rationale & Comments:
To find the start point of the AP, we begin from the peak and then go back in the trace until we find a derivative lower than 4mV/s. We repeat it for each spike.

Figure start_timing:

PSEUDOCODE
Calculate Derivative
For i=0;i<size(pk_timing);i++
    indexStart = pk_timing(i)
    While (derivativ(indexStart)>4)
        indexStart--
    end
    start_timing(i) = indexStart +1
end

EXAMPLE CODE
Too long
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA05</td>
<td>Start Value</td>
<td>[1xN float]</td>
<td>Primary Feature. Array of the start voltage for each spike</td>
<td></td>
</tr>
</tbody>
</table>

Rationale & Comments:

**PSEUDOCODE**
ap_start = Voltage(start_timing)

**EXAMPLE CODE**
```c
void AP_start(float * AP_start, float * Voltage, int AP_leng, double *AP_start_timing)
{
    int i,index;
    for (i=0;i<AP_leng;i++)
    {
        index = AP_start_timing[i];
        AP_start[i] = Voltage[index];
    }
}
```
### Short Name
PA06

### Full Name
End Time

### Format
[1xN double]

### Description
Primary Feature. Array of the end time value for each spike

### Rationale & Comments:
As to find the time of the start of an AP, we begin at peak and go forward in the trace until we find a derivative over than -2mV/s. Then we do the same for all AP.

#### Figure end_timing:

![end_timing](image)

#### PSEUDOCODE

```plaintext
Calculate Derivative
For i=0;i<size(pk_timing);i++
    indexStart = pk_timing(i)
    While (Derivative(indexStart)<-2)
        indexStart++
    end
end
end_timing(i) = indexStart -1
```

#### EXAMPLE CODE

Too long
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA07</td>
<td>End value</td>
<td>[1xN float]</td>
<td>Primary Feature. Array of the end value for each spike</td>
<td></td>
</tr>
</tbody>
</table>

Rationale & Comments:

**Figure ap_end:**

![End Point](image)

**PSEUDOCODE**

```plaintext
ap_end = Voltage(end_timing)
```

**EXAMPLE CODE**

```c
void AP_end(float * AP_end, float * Voltage, int AP_leng, double *AP_end_timing)
{
    int i, index;
    for (i=0; i<AP_leng; i++)
    {
        index = AP_end_timing[i];
        AP_end[i] = Voltage[index];
    }
}
```
Short Name | Full Name | Format | Description | Related E-code scripts
--- | --- | --- | --- | ---
PA08 | Ahp time | [1xM double] | Primary Feature. Array of the time value of each ahp | Figure ahp_depth_timing:

Rationale & Comments:
The ahp time of an AP is the time of the minimum voltage between the end time of the AP and the start time of the next one. In order to really get hyperpolarization, the minimum value should be lower than the start value of the AP.

We ignore the last ahp, in fact there is no new start time after the end of the AP.

**PSEUDO CODE**
For i=0;size(pk_timing);i++
   ahp_time = time of min Voltage between end_timing(i) and start_timing(i+1)
end

**EXAMPLE CODE**

```c
void AHP_depths_timing(double * valley_timing, float * Voltage, double * AP_start_timing, double * AP_end_timing, int num_spikes)
{
    int i,index,min_ind;
    double min_val, max_val;

    for (i=0; i<num_spikes-1;i++)
    {
        min_ind = min(Voltage, AP_end_timing[i], AP_start_timing[i+1]);
        index = AP_start_timing[i];
        min_val = Voltage[min_ind];
        max_val = Voltage[index];
        if (min_val <= max_val)
        {
            valley_timing[i] = min(Voltage, AP_end_timing[i], AP_start_timing[i+1]);
        }
        else
        {
            valley_timing[i] = 0;
        }
```
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA09</td>
<td>Ahp value</td>
<td>[1xM float]</td>
<td>Primary Feature. Array of the ahp value for each spike</td>
<td></td>
</tr>
</tbody>
</table>

Rationale & Comments:

![Figure ahp_depth:](image)

**PSEUDOCODE**

`ahp_depths = Voltage(ahp_depths_timing)`

**EXAMPLE CODE**

```c
void AHP_depths(float *ahp_depths, float *Voltage, int num_spikes, double *ahp_depths_timing)
{
    int i, index;
    for (i=0; i<num_spikes-1; i++)
    {
        index = ahp_depths_timing[i];
        ahp_depths[i] = Voltage[index];
    }
}
```
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA10</td>
<td>Mean Voltage</td>
<td>float</td>
<td>Secondary Feature. Float giving the mean value of the voltage during the stimulus</td>
<td>Figure v_mean :</td>
</tr>
</tbody>
</table>

Rationale & Comments: 

**PSEUDOCODE**

v_mean = mean(Voltage, start_stim, end_stim)

**EXAMPLE CODE**

```c
float meanF(float * input, int startIndex, int endIndex)
{
    float meanF;
    float sum, compt;
    int i;
    sum = 0;
    compt = 0;
    for(i=startIndex;i<endIndex;i++)
    {
        if(input[i]!=0){
            sum = sum + input[i];
            compt++;
        }
    }
    meanF = sum/(float)compt;
    return meanF;
}
```
Short Name | Full Name | Format | Description | Related E-code
--- | --- | --- | --- | ---
PA11 | Mean resting Potential | Float | Secondary Feature, Float giving the mean value of the voltage before the stimulus | scripts

Rationale & Comments:
This mean voltage corresponds to the resting potential of the cell before the stimulus. This resting potential is fixed by the experimentalist.

Figure v_rest:

PSEUDOCODE

```cpp
v_rest = mean(Voltage, 0, start_stim)
```

EXAMPLE CODE

```cpp
float meanF(float * input, int startIndex, int endIndex)
{
    float meanF;
    float sum, compt;
    int i;
    sum = 0;
    compt = 0;
    for(i=startIndex;i<endIndex;i++)
    {
        if(input[i]!=0)
        {
            sum = sum + input[i];
            compt++;
        }
    }
    meanF = sum/(float)compt;
    return meanF;
}
```
**III. PL FEATURES**

A PL Features currently implemented

<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL01</td>
<td>Access Resistance</td>
<td>float</td>
<td>Primary Feature. Float value corresponding to the access resistance of the pipette</td>
<td>Figure accRest:</td>
</tr>
</tbody>
</table>

Rationale & Comments:
The function to calculate the access resistance has been found thanks to tests on many data. There is an offset equal to 14.5 M\* which probably comes from the amplifier

**PSEUDOCODE**

accRest = 1.05 * (pk_hyperpol/ampStim) + 14.5

```plaintext
ampStim

accRest = 1.05*(1st_hyperpol / ampStim) + 14.5
```

**EXAMPLE CODE**

```plaintext
float acc_rest(float* Voltage, float* Current, double startStim, double endStim) {
    int i;
    int indexStart;
    float ampStim;
    float accRest;
    float factor;
    indexStart = startStim;
    i = startStim;
    while (Voltage[i+1]-Voltage[i] < 0) {
        i++;
    }
    ampStim = amp_stim(Current, startStim, endStim);
    if (Voltage[i] < 1) {
        factor = 1000;
    } else {
        factor = 1000000;
    }
    accRest = ((Voltage[indexStart] - Voltage[i])/ampStim)*factor*1.04 + 14.5;
    return accRest;
}
```
**Short Name** | **Full Name** | **Format** | **Description** | **Related E-code scripts**
---|---|---|---|---
PL02 | Time membrane | [1x3 float] | Primary Feature. Return the parameter of the exponential fitting of for the time membrane constant | 

**Rationale & Comments:**
The function returns the parameters a, b, and c to fit the curve with the following function:

\[ f = c + a \cdot e^{b/t} \]

**PSEUDOCODE**
For the moment the value of c is not well defined, there was a problem with the derivative, but it still gives good results.
It calculated first b thanks to the derivative, then a and finally c.

**EXAMPLE CODE**
```c
void membrane_time(float *param, float *Voltage, int volt_leng, double dt)
{
    float a, b, c;
    int i, start, end;
    int count = 0;
    b = 0;
    a = 0;
    c = 0;
    start = 0.025/dt;
    end = 0.25/dt;
    for (i=start;i<end-10;i++)
    {
        b = b + log(Voltage[i+10]/Voltage[i]) / (1/(i+10)*dt) - 1/(i*dt));
        count++;
    }
    b = b/count;
    for (i=start;i<end;i++)
    {
        a = a + (Voltage[i+10] - Voltage[i]) / (exp(b/((i+10)*dt)) - exp(b/(i*dt)));
    }
    a = a/count;
    for (i=start;i<end;i++)
    {
        c = c + Voltage[i] - a * exp(b/(i*dt));
    }
    c = c/count;
    param[0] = a;
    param[1] = b;
    param[2] = c;
}
```
### Short Name Full Name Format Description Related E-code scripts

| PL03 | Input Resistance | [1x2 float] | Primary Feature. Return the two input resistance calculated on IV traces | Figure membrane_time: |

**Rationale & Comments:**
Calculates the input resistance based on all the IV traces and takes the means values one for the first part of the trace and one for the second. The two resistances are the steady one and the peak one.

**PSEUDOCODE**
It first takes the index of the mean point on the first trace, and calculates the resistance around this time by taking the mean voltage and mean current. Then it takes the mean after the middle time until the end of the stimulus to calculate the steady resistance. The times values have been fixed with experimentalist

**EXAMPLE CODE**
```c
void in_res(float* In_res, float *Voltage, float* Current, int point_length, int nbTraces, double dt) {
    int j;
    int indexMin, indexSS, indexES, indexSP, indexEP;
    float res_peak_temp = 0;
    float res_steady_temp = 0;
    float VoltageMeanP1, CurrentMeanP1, VoltageMeanS1, CurrentMeanS1,
    VoltageMeanP2, CurrentMeanP2, VoltageMeanS2, CurrentMeanS2;

    indexMin = min(Voltage,0,point_length);
    indexSP = indexMin - 0.005/dt;
    indexEP = indexMin + 0.005/dt;
    indexSS = 0.5/dt;
    indexES = 0.95/dt;

    for (j=0;j<nbTraces-1;j++)
    {
        VoltageMeanP1 = meanF(Voltage, indexSP+j*point_length, indexEP+j*point_length);
        CurrentMeanP1 = meanF(Current, indexSP+j*point_length, indexEP+j*point_length);
        VoltageMeanS1 = meanF(Voltage, indexSS+j*point_length, indexES+j*point_length);
        CurrentMeanS1 = meanF(Current, indexSS+j*point_length, indexES+j*point_length);
        VoltageMeanP2 = meanF(Voltage, indexSS+(j+1)*point_length,
        indexEP+(j+1)*point_length);
        CurrentMeanP2 = meanF(Current, indexSS+(j+1)*point_length,
        indexEP+(j+1)*point_length);
        VoltageMeanS2 = meanF(Voltage, indexSS+(j+1)*point_length,
        indexES+(j+1)*point_length);
        CurrentMeanS2 = meanF(Current, indexSS+(j+1)*point_length,
        indexES+(j+1)*point_length);

        res_steady_temp = res_steady_temp + ((VoltageMeanS2 - VoltageMeanS1)/(CurrentMeanS2 - CurrentMeanS1));
        res_peak_temp = res_peak_temp + ((VoltageMeanP2 - VoltageMeanP1)/(CurrentMeanP2 - CurrentMeanP1));
    }
    In_res[0] = res_peak_temp/nbTraces;
    In_res[1] = res_steady_temp/nbTraces;
}
```
IV. SA FEATURES
A. SA features currently implemented

<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA01</td>
<td>Rising time</td>
<td>[1xN double]</td>
<td>Secondary Feature. Array of the time of rising part for each spikes.</td>
<td></td>
</tr>
</tbody>
</table>

Rationale & Comments:
The rising part corresponds to the part between the start point and the peak point

![Rise part](image)

**PSEUDOCODE**

```plaintext```
For i=0; i<size(Ap_pk); i++
   rise_time(i) = pk_timing(i) - start_timing(i)
end
```

**EXAMPLE CODE**

```c
void time(double * time, double * start_time, double * end_time, int AP_leng)
{
   int i;
   for(i=0; i<AP_leng; i++)
   {
      time[i] = end_time[i] - start_time[i];
   }
}
```
Short Name | Full Name | Format | Description | Related E-code scripts
--- | --- | --- | --- | ---
SA02 | falling time | [1xN double] | Secondary Feature. Array of the time of falling part for each spikes. |

Rationale & Comments:
The falling part corresponds to the part between the peak point and the start point

**Figure fall_time**:
![Graph showing the falling part](image)

**PSEUDOCODE**
For i=0;i<size(Ap_pk);i++
fall_time(i) = end_timing(i) – pk_timing(i)
end

**EXAMPLE CODE**
```c
void time(double * time, double * start_time, double * end_time, int AP_leng)
{
    int i;
    for(i=0;i<AP_leng;i++)
    { 
         time[i] = end_time[i] - start_time[i];
    }
}
```
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA03</td>
<td>Rise slope</td>
<td>[1xN float]</td>
<td>Secondary Feature. Array of the slope of the rising part for each spikes.</td>
<td>scripts</td>
</tr>
</tbody>
</table>

Rationale & Comments:
The rising part corresponds to the part between the start point and the peak point.

Figure rise_slope:

![Rise part](image)

PSEUDOCODE
For i=0;i<size(Ap_pk);i++
    rise_slope[i] = rise_time[i] / (ap_pk[i] - ap_start[i])
end

EXAMPLE CODE
```c
void slope(float *slope, double *time, float *start_point_value, float *end_point_value, int AP_leng, double dt) {
    int i;
    int factor=1;
    for(i=0;i<AP_leng;i++)
    {
        if (start_point_value[i]<-1 || end_point_value[i]<-1)
        {
            factor = 1000; /*to adapt to mV or V*/
        }
        slope[i] = (end_point_value[i]-start_point_value[i])/(time[i]*dt)/factor;
    }
}
```
Short Name | Full Name | Format Description Related E-code scripts
---|---|---|---
SA04 | Fall slope | [1xN float] | Secondary Feature. Array of the slope of falling part for each spikes.

Rationale & Comments:
The falling part corresponds to the part between the peak point and the start point.

**Figure fall_slope**:

![Diagram of fall_slope](image)

**PSEUDOCODE**
For i=0;i<size(Ap_pk);i++
fall_slope(i) = fall_time(i) / (ap_end(i) - ap_pk(i))
end

**EXAMPLE CODE**
```c
void slope(float * slope, double * time, float * start_point_value, float * end_point_value, int AP_leng, double dt)
{
    int i;
    int factor=1;
    for(i=0;i<AP_leng;i++)
    {
        if (start_point_value[i]<-1 || end_point_value[i]<-1)
        {
            factor = 1000; /*to adapt to mV or V */
        }
        slope[i] = (end_point_value[i]-start_point_value[i])/(time[i]*dt)/factor;
    }
}
```
Short Name | Full Name | Format | Description | Related E-code scripts
--- | --- | --- | --- | ---
SA05 | Ahp delay | [1xN-1 double] | Secondary Feature. Array of the delay of each ahp | Figure ahp_delay:

Rationale & Comments:
The delay corresponds to the time between the end of the spike and the ahp. It could be useful in order to have an idea of the shape of the end of the AP

**PSEUDOCODE**

```plaintext
For i=0;i<size(ahp_timing);i++
   ahp_delay = ahp_timing(i) – end_timing(i)
end
```

**EXAMPLE CODE**

```c
void AHP_delay(double * AHP_delay, double * AP_end_timing, double * ahp_timing, int AP_leng)
{
    int i;
    for(i=0;i<AP_leng-1;i++)
    {
        if(ahp_timing[i]!=0)
        {
            AHP_delay[i] = ahp_timing[i] - AP_end_timing[i];
        }
        else
        {
            AHP_delay[i] = 0;
        }
    }
}
```
SA06   |  Ahp slope  |  [1xN-1 float]  |
|-------|-------------|----------------|

**Rationale & Comments:**
Ahp slope corresponds to the slope between the ahp of an AP and the start of the next AP. There is no slope for the last AP, if fact we ignore the last ahp.

**PSEUDOCODE**
For i=0;i<size(ahp_timing);i++
   ahp_slope[i] = (start_timing[i+1] - ahp_timing[i]) / (ap_start[i+1] - ahp_depths[i])
end

**EXAMPLE CODE**
```c
void ahp_slope(float * ahp_slope, double * AHP_depths_timing, float * AHP_depths,
               double * AP_start_timing, float * AP_start, int AP_leng, double dt)
{
    int i;
    int factor=1;

    for(i=0;i<AP_leng-1;i++)
    {
        if(AHP_depths_timing[i] != 0)
        {
            if (AP_start[i]<-1) /*to adapt to mV or V */
                factor = 1000;
            ahp_slope[i] = (AP_start[i+1] - AHP_depths[i])/(AP_start_timing[i+1]-AHP_depths_timing[i])*dt / factor;
        }
        else
            ahp_slope[i] = 0;
    }

    ahp_slope[AP_leng-1] = 0;
}
```

**Description**
Secondary Feature. Array of the slope after each ahp

**Figure fall_slope :**
![Ahp slope](image)
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA07</td>
<td>Inter spikes interval</td>
<td>[1xN-1 double]</td>
<td>Secondary Feature. Array of the inter spikes interval.</td>
<td></td>
</tr>
</tbody>
</table>

**Rationale & Comments:**
The ISI corresponds to time between each AP, this time is calculated between the peak of the APs.

**Figure ISI:**
![ISI figure](image)

**PSEUDOCODE**
For i=0;i<size(ap_pk)-1;i++
    ISI(i) = pk_timing(i+1) – pk_timing(i)
end

**EXAMPLE CODE**
```c
void calc_isi (double * ISI, double * AP_peaks_timing, int AP_leng) {
    int i;
    for(i=0; i<AP_leng-1;i++)
    {
        ISI[i] = AP_peaks_timing[i+1] - AP_peaks_timing[i];
    }
}
```
SA09  Isi rate on initial part to middle part

Rationale & Comments:
This feature is calculated if the trace contains at least 10 AP.
Initial means the 2 first APs, and middle means from the 3rd AP to the 2/3 AP.
Could be useful to see if there is a burst.

PSEUDOCODE

```plaintext
isi_init = (isi[0]+isi[1])/2
isi_middle = mean(isi, 2, length(isi)*2/3)
isi_initial_to_middle = isi_init/isi_middle
```

EXAMPLE CODE

```c
float ISI_initial_to_middle(double *ISI, int ISI_leng)
{
    float ISI_initial_to_middle;
    int index, i;
    double sum;
    float mean1, mean2;
    if(ISI_leng>9)
    {
        index = floor(ISI_leng*2/3);
        mean1 = (ISI[0]+ISI[1])/2;
        sum=0;
        for(i=2;i<index;i++)
        {
            sum = sum + ISI[i];
        }
        mean2 = sum/(index-2);
        ISI_initial_to_middle = mean1/mean2;
    }
    else
    {
        ISI_initial_to_middle = 0;
    }

    return ISI_initial_to_middle;
}
```
Short Name | Full Name | Format | Description | Related E-code scripts
---|---|---|---|---
SA10l | Isi rate on middle part to final part | float | Secondary Feature. Float giving the rate of the mean of the middle ISIs on the mean of the final ISIs | Figure:

**Rationale & Comments:**
This feature is calculated if the trace contains at least 10 AP. Middle means form the 3rd AP to the 2/3 AP and final means the last third part. Could be useful to see if there is adapting behavior or not

**PSEUDOCODE**

```plaintext
isi_middle = mean(isi, 2, length(isi)*2/3) 
isi_final = mean(isi, length(isi)*2/3, length(isi)) 
isi_middle_to_final = isi_middle / isi_final 
```

**EXAMPLE CODE**

```c
float ISI_middle_to_final(double *ISI, int ISI_leng) 
{
    float ISI_middle_to_final; 
    int index, i; 
    double sum; 
    float mean1, mean2; 
    if(ISI_leng>9) 
    {
        index = floor(ISI_leng*2/3); 
        sum=0; 
        for(i=2;i<index;i++) 
        {
            sum = sum + ISI[i]; 
        }
        mean1 = sum/(index-2); 
        sum=0; 
        for(i=index;i<ISI_leng;i++) 
        {
            sum = sum + ISI[i]; 
        }
        mean2 = sum/(ISI_leng-index);
        ISI_middle_to_final = mean1/mean2; 
    } 
    else 
    { 
        ISI_middle_to_final = 0; 
    }
    return ISI_middle_to_final; 
}
```
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA11</td>
<td>ISI Increasing</td>
<td>float</td>
<td>Secondary Feature.</td>
<td>scripts</td>
</tr>
<tr>
<td></td>
<td>percentage</td>
<td></td>
<td>Float giving the percentage of ISI increasing</td>
<td></td>
</tr>
</tbody>
</table>

Rationale & Comments:
Takes the mean ISI on the three first APs, and compares it to the mean of the last five APs.

**PSEUDOCODE**

IsiStart = (isi(0)+isi(1))/2
isiEnd = mean(isi,length-5,length)
percentage = (isiEnd - isiStart) / isiEnd * 100

**EXAMPLE CODE**

    float ISI_percentage(double *ISI, int ISI_leng)
    {
        float ISI_pourcentage;
        float mean1, mean2;
        double sum;
        int i;

        if(ISI_leng>9)
        {
            mean1 = (ISI[0] + ISI[1])/2;
            sum = 0;
            for (i=ISI_leng-5;i<ISI_leng;i++)
            {
                sum = sum + ISI[i];
            }
            mean2 = sum/5;
            ISI_pourcentage = (mean2-mean1)/mean2*100;
        }
        else
        {
            ISI_pourcentage = 0;
        }

        return ISI_pourcentage;
    }
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA12</td>
<td>frequency</td>
<td>float</td>
<td>Secondary Feature. Float giving the firing frequency</td>
<td></td>
</tr>
</tbody>
</table>

Rationale & Comments:

Figure:

**PSEUDOCODE**

\[
F = \frac{\text{size(\text{ap}_{pk})}}{(\text{end}_{stim} - \text{start}_{stim})}
\]

**EXAMPLE CODE**

```c
float frequency(int AP_leng, double startStim, double endStim, double dt)
{
    float frequency;
    frequency = AP_leng/((endStim-startStim)*dt);
    return frequency;
}
```
SA13  AP drop  float

**Short Name**  **Full Name**  **Format**  **Description**  **Related E-code**

Secondary Feature. Float giving the difference between the height of the 2 first APs

**Rationale & Comments:**

Figure ap_drop:

```
**PSEUDOCODE**
ap_drop = ap_pk(1) – ap_pk(0)

**EXAMPLE CODE**
float AP_drop(float * AP_pk, int AP_leng)
{
    float AP_drop;
    if (AP_leng>1)
    {
        AP_drop = AP_pk[1]-AP_pk[0];
    }
    else
    {
        AP_drop = 0;
    }
    return AP_drop;
}
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA14</td>
<td>Mean width</td>
<td>float</td>
<td>Secondary Feature. Float giving the mean value of the widths of each AP</td>
<td>scripts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Figure ap_width_mean :</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The width is taken at the half amplitude</td>
<td></td>
</tr>
</tbody>
</table>

**Rationale & Comments:**

**PSEUDOCODE**

```plaintext
ap_width_mean = mean(ap_width)
```

**EXAMPLE CODE**

```c
float meanD(double * input, int startIndex, int endIndex)
{
    double meanD;
    double sum, compt;
    int i;
    sum = 0;
    compt = 0;
    for(i=startIndex;i<endIndex;i++)
    {
        if(input[i]!=0){
            sum = sum + input[i];
            compt++;
        }
    }
    meanD = sum/(float)compt;

    return meanD;
}
```
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA15</td>
<td>Mean of Height AP</td>
<td>float</td>
<td>Secondary Feature. Float giving the mean value of the heights of each AP</td>
<td></td>
</tr>
</tbody>
</table>

**Rationale & Comments:**

Figure `ap_height_mean`:

![Ap height mean](image)

**PSEUDOCODE**

```
ap_height_mean = mean(ap_pk)
```

**EXAMPLE CODE**

```c
float meanF(float *input, int startIndex, int endIndex)
{
    float meanF;
    float sum, compt;
    int i;
    sum = 0;
    compt = 0;
    for(i=startIndex; i<endIndex; i++)
    {
        if(input[i]!=0)
        {
            sum = sum + input[i];
            compt++;
        }
    }
    meanF = sum/(float)compt;
    return meanF;
}
```
SA16  Mean rising time  float  Secondary Feature. Float giving the mean of the rise times

Rationale & Comments:

Figure

PSEUDOCODE

rise_time_mean = mean(rise_time)

EXAMPLE CODE

float meanD(double * input, int startIndex, int endIndex)
{
    double meanD;
    double sum, compt;
    int i;
    sum = 0;
    compt = 0;
    for(i=startIndex;i<endIndex;i++)
    {
        if(input[i]!=0)
        {
            sum = sum + input[i];
            compt++;
        }
    }
    meanD = sum/(float)compt;

    return meanD;
}
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA17</td>
<td>Mean falling time</td>
<td>float</td>
<td>Secondary Feature. Float giving the mean of the fall times</td>
<td>scripts</td>
</tr>
</tbody>
</table>

**Rationale & Comments:**

**PSEUDOCODE**

```plaintext
fall_time_mean = mean(fall_time)
```

**EXAMPLE CODE**

```c
float meanD(double * input, int startIndex, int endIndex)
{
    double meanD;
    double sum, compt;
    int i;
    sum = 0;
    compt = 0;
    for(i=startIndex;i<endIndex;i++)
    {
        if(input[i]!=0){
            sum = sum + input[i];
            compt++;
        }
    }
    meanD = sum/(float)compt;
    return meanD;
}
```
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA18</td>
<td>Rise slope mean</td>
<td>float</td>
<td>Secondary Feature. Float giving the mean of the rise slopes</td>
<td>scripts</td>
</tr>
</tbody>
</table>

Rationale & Comments:

Figure:

**PSEUDOCODE**

```c
rise_slope_mean = mean(rise_slope)
```

**EXAMPLE CODE**

```c
float meanF(float * input, int startIndex, int endIndex)
{
    float meanF;
    float sum, compt;
    int i;
    sum = 0;
    compt = 0;
    for(i=startIndex;i<endIndex;i++)
    {
        if(input[i]!=0)
        {
            sum = sum + input[i];
            compt++;
        }
    }
    meanF = sum / (float) compt;
    return meanF;
}
```
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA19</td>
<td>Fall slope mean</td>
<td>float</td>
<td>Secondary Feature. Float giving the mean of the fall slopes</td>
<td></td>
</tr>
</tbody>
</table>

**Rationale & Comments:**

**Figure:**

**PSEUDOCODE**

```plaintext
fall_slope_mean = mean(fall_slope)
```

**EXAMPLE CODE**

```c
float meanF(float * input, int startIndex, int endIndex)
{
    float meanF;
    float sum, compt;
    int i;
    sum = 0;
    compt = 0;
    for(i=startIndex; i<endIndex; i++)
    {
        if(input[i]! = 0)
        {
            sum = sum + input[i];
            compt++;
        }
    }
    meanF = sum/(float)compt;

    return meanF;
}
```
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA20</td>
<td>Mean of the ahp delay</td>
<td>float</td>
<td>Secondary Feature. Float giving the mean of the ahp delays</td>
<td></td>
</tr>
</tbody>
</table>

Rationale & Comments:

Figure:

**PSEUDOCODE**

```plaintext
ahp_delay_mean = mean(ahp_delay)
```

**EXAMPLE CODE**

```c
float meanD(double * input, int startIndex, int endIndex)
{
    double meanD;
    double sum, compt;
    int i;
    sum = 0;
    compt = 0;
    for(i=startIndex;i<endIndex;i++)
    {
        if(input[i]!=0){
            sum = sum + input[i];
            compt++;
        }
    }
    meanD = sum/(float)compt;
    return meanD;
}
```
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code</th>
<th>scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA21</td>
<td>Mean ahp value</td>
<td>float</td>
<td>Secondary Feature. Float giving the mean of the ahp values</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Rationale & Comments:**

Figure:

```
PESEDOCODE
ahp_depths_mean = mean(ahp_depths)

EXAMPLE CODE
float AHP_depths_mean(float * AHP_depths, double * AHP_timing, int AP_leng) {
    int i,sum,compt;
    float AHP_depths_mean;
    sum = 0;
    compt = 0;
    for(i=0;i<AP_leng-1;i++){
        if(AHP_timing[i]!=0){
            sum = sum + AHP_depths[i];
            compt++;
        }
    }

    AHP_depths_mean = sum/(float)compt;

    return AHP_depths_mean;
}
```
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA22</td>
<td>Ahp slope mean</td>
<td>float</td>
<td>Secondary Feature. Float giving the mean of the ahp slopes</td>
<td></td>
</tr>
</tbody>
</table>

Rationale & Comments:

**Example Code**

```c
float meanF(float * input, int startIndex, int endIndex) {
    float meanF;
    float sum, compt;
    int i;
    sum = 0;
    compt = 0;
    for(i=startIndex;i<endIndex;i++)
    {    
        if(input[i]!=0)
        {    
            sum = sum + input[i];
            compt++;
        }
    }
    meanF = sum/(float)compt;
    return meanF;
}
```
<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA23</td>
<td>Adapting index</td>
<td>float</td>
<td>Secondary Feature. float giving the adaptation of the trace concerning the firing behavior</td>
<td></td>
</tr>
</tbody>
</table>

Rationale & Comments:
This feature is useful to differentiate adaptation and fast spiking behavior

**PSEUDOCODE**
calculate the difference between each consecutive ISI and divide it by the sum of each. Then the global result is divided by the number of ISI.

**EXAMPLE CODE**
```c
float adapt_ind(double *ISI, int ISI_leng)
{
    int i;
    float tmpVal = 0;
    float adaptInd;
    
    for (i=1;i<ISI_leng;i++){
        tmpVal = tmpVal + (ISI[i] - ISI[i-1])/(ISI[i] + ISI[i-1]);
    }
    adaptInd = tmpVal/ISI_leng;
    return adaptInd;
}
```

B. SA features under consideration

<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA24</td>
<td>AP gradient</td>
<td>[1xN-1 double]</td>
<td>Secondary Feature. double giving the gradient of the peaks values of the APs</td>
<td></td>
</tr>
</tbody>
</table>

Rationale & Comments:
This gradient vector could allow to determine if the value of the peaks of the AP decrease too much along the time (could be use for the quality check)

**PSEUDOCODE**
ap_grad = diff(ap_pk)

**EXAMPLE CODE**
Not implemented for the moment
V. SL FEATURES
A. SL FEATURES Currently implemented

<table>
<thead>
<tr>
<th>Short Name</th>
<th>Full Name</th>
<th>Format</th>
<th>Description</th>
<th>Related E-code Scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>SL01</td>
<td>First Spike delay</td>
<td>double</td>
<td>Secondary Feature. double giving the delay of the first spike</td>
<td></td>
</tr>
</tbody>
</table>

Rationale & Comments:
This delay corresponds to the time between the start time of the stimulus and the start time of the first AP

Figure first_sp_delay:

![First spike delay](image)

**PSEUDOCODE**

first_sp_delay = start_timing(0) - start_stim

**EXAMPLE CODE**

double first_spike_delay(double startStim, double *AP_start_timing)
{
    double first_spike_delay;
    first_spike_delay = AP_start_timing[0] - startStim;
    return first_spike_delay;
}
VI. APPLICATION

The features functions are all joined in a mex file, which can be launched automatically within a .m file or by clicking on a button on the matlab data browser (see data browser user Guide document). This mex file creates two arrays and one structure. The first one is an array of structure concerning the APs, its length corresponds to the number of AP and each structures contains the key points of an AP (start, peak, width, end and ahp but also the rise and fall time and slope), it is called “AP”. The second array concerns the Inter Spikes Intervals, it is called “ISI”. The last structure, called “Features”, gives the values of some secondary features.

This features extraction has three goals:

i. The first one is to realize Quality Check program able to say whether the experiment is good or not. It means to check if the cell has a good behavior or not.

ii. The second one is to implement a classifier, able to classify traces according the Petilla table. It will mainly use the structure “Features” to train the classifier.

iii. The third concerns the calibration. The extraction features have to be done automatically on a huge set of traces in order to have good statistics and have relevant features to fit the simulated cells.
Appendix 3 - Quality Check

Bad traces analysis:

A. First look at the Access Resistance (should be < 30MΩ)
B. Pay attention to some artifacts within the spikes (should take a mean value of the high spikes and compare each spike value to this, if the difference is over a given threshold then remove the trace)
C. If the mean of the V start is less than V_rest then it's a bad cell.
D. The value at the end of the record should be around the V_rest value, otherwise it means that the cell is still depolarize even after the end of the stimulus.
E. Look at the evolution of the value of the plateau, if it decreases to much, the cell is bad.
F. It can happen that the cell fires before the stimulus or after it, then it is important to look at the times of the first and last spike.
G. Then there are some shape parameters like the AP height which must be over -10mV and the AP width which must be lower 5 ms.

Quality Check Criteria :

a) AP > -10mV
b) -60mV < V_rest<-80mV
c) width < 5ms
d) ahp_depth > 5 * std(ahp_depth).
e) Access Resistance < 33MΩ
f) mean start voltage > resting potential
g) 0.9 * resting potential< end voltage < 1.1 * resting potential
h) 1st_AP_start_time < start_stim && last_start_time < end_stim

Quality Check Implementation:

Three scripts concerning Quality Check have been written. They are all on the svn in the directory /analysis/trunk/DataBrowser/TraceGUI.
The first one is “QualityCheckV2.m”. This function extracts the features of the traces of one cell and then looks at the criteria written above. For each criteria the trace has 0 or 1 depending if it is OK or nor. Finally the function do the mean for each criteria on all the trace and return for each criteria the percentage of good traces for the given cell. Then the user can select for example just the cells where all the traces have an access resistance lower than 33MΩ.
The second one is “QualityCheckOneTrace.m”, this one checks just the quality of one trace and not of all the trace of a cell. This could be useful for selecting the good traces of a cell and removing the bad one, to do some statistics for example.
The last one is “automaticQualityCheck.m”, it goes through all the cells of the database and check their quality, then it returns the result in a matlab structure order by metype, then etype and finally cellnames. This function also plots a graph with the percentage of bad cells per
criteria.
Appendix 4 – Features Analysis

This appendix presents the different results of the features analysis. The first set of graphs concerns the fitness features, the second one the generalization features, the third one the influence of the ISI features and finally the last one the features chosen for the classifier.
## Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>45</td>
<td>Fitness cAD-bAD</td>
<td>117</td>
</tr>
<tr>
<td>46</td>
<td>Fitness cAD-cFS</td>
<td>118</td>
</tr>
<tr>
<td>47</td>
<td>Fitness cAD-dFS</td>
<td>118</td>
</tr>
<tr>
<td>48</td>
<td>Fitness cAD-cST</td>
<td>119</td>
</tr>
<tr>
<td>49</td>
<td>Fitness cAD-bIS</td>
<td>119</td>
</tr>
<tr>
<td>50</td>
<td>Fitness cAD-cNA</td>
<td>120</td>
</tr>
<tr>
<td>51</td>
<td>Fitness bAD-cFS</td>
<td>120</td>
</tr>
<tr>
<td>52</td>
<td>Fitness bAD-dFS</td>
<td>121</td>
</tr>
<tr>
<td>53</td>
<td>Fitness bAD-cST</td>
<td>121</td>
</tr>
<tr>
<td>54</td>
<td>Fitness bAD-bIS</td>
<td>122</td>
</tr>
<tr>
<td>55</td>
<td>Fitness cFS-dFS</td>
<td>122</td>
</tr>
<tr>
<td>56</td>
<td>Fitness cFS-cST</td>
<td>123</td>
</tr>
<tr>
<td>57</td>
<td>Fitness cFS-bIS</td>
<td>123</td>
</tr>
<tr>
<td>58</td>
<td>Fitness cFS-cNA</td>
<td>124</td>
</tr>
<tr>
<td>59</td>
<td>Fitness dFS-cST</td>
<td>124</td>
</tr>
<tr>
<td>60</td>
<td>Fitness dFS-bIS</td>
<td>125</td>
</tr>
<tr>
<td>61</td>
<td>Fitness dFS-cNA</td>
<td>125</td>
</tr>
<tr>
<td>62</td>
<td>Fitness cST-bIS</td>
<td>126</td>
</tr>
<tr>
<td>63</td>
<td>Fitness cST-cNA</td>
<td>126</td>
</tr>
<tr>
<td>64</td>
<td>Fitness bIS-cNA</td>
<td>127</td>
</tr>
<tr>
<td>65</td>
<td>Generalization cAD-bAD</td>
<td>128</td>
</tr>
<tr>
<td>66</td>
<td>Generalization cAD-cFS</td>
<td>129</td>
</tr>
<tr>
<td>67</td>
<td>Generalization cAD-dFS</td>
<td>129</td>
</tr>
<tr>
<td>68</td>
<td>Generalization cAD-cST</td>
<td>130</td>
</tr>
<tr>
<td>69</td>
<td>Generalization cAD-bIS</td>
<td>130</td>
</tr>
<tr>
<td>70</td>
<td>Generalization cAD-cNA</td>
<td>131</td>
</tr>
<tr>
<td>71</td>
<td>Generalization bAD-cFS</td>
<td>131</td>
</tr>
<tr>
<td>72</td>
<td>Generalization bAD-dFS</td>
<td>132</td>
</tr>
<tr>
<td>73</td>
<td>Generalization bAD-cST</td>
<td>132</td>
</tr>
<tr>
<td>74</td>
<td>Generalization bAD-bIS</td>
<td>133</td>
</tr>
<tr>
<td>75</td>
<td>Generalization bAD-cNA</td>
<td>133</td>
</tr>
<tr>
<td>76</td>
<td>Generalization cFS-dFS</td>
<td>134</td>
</tr>
<tr>
<td>77</td>
<td>Generalization cFS-cST</td>
<td>134</td>
</tr>
<tr>
<td>78</td>
<td>Generalization cFS-bIS</td>
<td>135</td>
</tr>
<tr>
<td>79</td>
<td>Generalization cFS-cNA</td>
<td>135</td>
</tr>
<tr>
<td>80</td>
<td>Generalization dFS-cST</td>
<td>136</td>
</tr>
<tr>
<td>81</td>
<td>Generalization dFS-cNA</td>
<td>136</td>
</tr>
<tr>
<td>82</td>
<td>Generalization cST-bIS</td>
<td>137</td>
</tr>
<tr>
<td>83</td>
<td>Generalization cST-cNA</td>
<td>137</td>
</tr>
<tr>
<td>84</td>
<td>Generalization bIS-cNA</td>
<td>138</td>
</tr>
<tr>
<td>85</td>
<td>ISI Features cAD-bAD</td>
<td>139</td>
</tr>
<tr>
<td>86</td>
<td>ISI Features cAD-cFS</td>
<td>139</td>
</tr>
<tr>
<td>87</td>
<td>ISI Features cAD-dFS</td>
<td>140</td>
</tr>
</tbody>
</table>
Fitness Features

Figure 45 Fitness cAD-bAD
Figure 46 Fitness cAD-cFS

Figure 47 Fitness cAD-dFS
Figure 50 Fitness cAD-cNA

Figure 51 Fitness bAD-cFS
Figure 52 Fitness bAD-dFS

Figure 53 Fitness bAD-cST
Figure 56 Fitness cFS-cST

Figure 57 Fitness cFS-bIS
Figure 58 Fitness cFS-cNA

Figure 59 Fitness dFS-cST
Figure 60 Fitness dFS-bIS

Figure 61 Fitness dFS-cNA
Figure 62 Fitness cST-bIS

Figure 63 Fitness cST-cNA
Figure 64 Fitness bIS-cNA
1. Generalization Features

Figure 65 Generalization cAD-bAD
Figure 66 Generalization cAD-cFS

Figure 67 Generalization cAD-dFS
Figure 68 Generalization cAD-cST

Figure 69 Generalization cAD-bIS
Figure 70 Generalization cAD-cNA

Figure 71 Generalization bAD-cFS
<table>
<thead>
<tr>
<th>V mean</th>
<th>V rest</th>
<th>Rise time</th>
<th>Fall time</th>
<th>Rise slope</th>
<th>Fall slope</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.5</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 72 Generalization bAD-dFS**

<table>
<thead>
<tr>
<th>V mean</th>
<th>V rest</th>
<th>Rise time</th>
<th>Fall time</th>
<th>Rise slope</th>
<th>Fall slope</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.5</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 73 Generalization bAD-cST**
Figure 74 Generalization bAD-bIS

Figure 75 Generalization bAD-cNA
Figure 76 Generalization cFS-dFS

Figure 77 Generalization cFS-cST
Figure 78 Generalization cFS-bIS

Figure 79 Generalization cFS-cNA
Figure 80 Generalization dFS-cST

Figure 81 Generalization dFS-cNA
Figure 82 Generalization cST-bIS

Figure 83 Generalization cST-cNA
Figure 84 Generalization bIS-cNA
2. ISI Features

**Figure 85 ISI Features cAD-bAD**

**Figure 86 ISI Features cAD-cFS**
<table>
<thead>
<tr>
<th>Feature</th>
<th>cAD</th>
<th>dFS</th>
<th>cAD</th>
<th>cST</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISI init middle</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>ISI middle final</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>ISI percentage</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Rise time</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>AHP delay</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>AHP slope</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

**Figure 87 ISI Features cAD-dFS**

<table>
<thead>
<tr>
<th>Feature</th>
<th>cAD</th>
<th>cST</th>
<th>cAD</th>
<th>dFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISI init middle</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>ISI middle final</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>ISI percentage</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Rise time</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>AHP delay</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>AHP slope</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

**Figure 88 ISI Features cAD-cST**
Figure 91 ISI Features bAD-cFS

Figure 92 ISI Features bAD-dFS
Figure 93 ISI Features bAD-cST

Figure 94 ISI Features bAD-bIS
Figure 95 ISI Features bAD-cNA

Figure 96 ISI Features cFS-dFS
Figure 97 ISI Features cFS-cST

Figure 98 ISI Features cFS-bIS
Figure 99 ISI Features cFS-cNA

Figure 100 ISI Features dFS-cST
Figure 101 ISI Features dFS-bIS

Figure 102 ISI Features cST-bIS
Figure 103 ISI Features dFS-cNA

Figure 104 ISI Features cST-cNA
Figure 105 ISI Features bIS-cNA
3. Classifier Features

Figure 106 Classifier Features bAD-bIS

Figure 107 Classifier Features bAD-cFS
Figure 108 Classifier Features bAD-cNA

Figure 109 Classifier Features bAD-cST
Figure 110 Classifier Features bAD-cFS

Figure 111 Classifier Features bIS-cNA
Figure 114 Classifier Features cAD-cNA

Figure 115 Classifier Features cAD-cST
Figure 116 Classifier Features cAD-dFS

Figure 117 Classifier Features dFS-bIS
Figure 118 Classifier Features cFS-cNA

Figure 119 Classifier Features cFS-cST
Figure 120 Classifier Features cFS-dFS

Figure 121 Classifier Features cST-bIS
Figure 124 Classifier Features $dFS-cNA$

Figure 125 Classifier Features $dFS-cST$
Appendix 5 – Classification

This appendix presents the different results of the classification. The part shows the one from the Matlab function “classify”, with pictures of the data points with border line of the classification. The second part shows the results of the knn function and the last one the result of the svm method.
Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>126</td>
<td>Classify Function Linear Distance</td>
<td>162</td>
</tr>
<tr>
<td>127</td>
<td>Classify Function Diaglinear Distance</td>
<td>162</td>
</tr>
<tr>
<td>128</td>
<td>Classify Function Diagquadratic Distance</td>
<td>162</td>
</tr>
<tr>
<td>129</td>
<td>Classify Function Mahalanobis Distance</td>
<td>162</td>
</tr>
<tr>
<td>130</td>
<td>Classify Function Quadratic Distance</td>
<td>163</td>
</tr>
<tr>
<td>131</td>
<td>Classify Function One Against All</td>
<td>163</td>
</tr>
<tr>
<td>132</td>
<td>Classify Separation Line bAS-bIS</td>
<td>163</td>
</tr>
<tr>
<td>133</td>
<td>Classify Separation Line bAD-cFS</td>
<td>164</td>
</tr>
<tr>
<td>134</td>
<td>Classify Separation Line bAD-cNA</td>
<td>164</td>
</tr>
<tr>
<td>135</td>
<td>Classify Separation Line bAd-cST</td>
<td>165</td>
</tr>
<tr>
<td>136</td>
<td>Classify Separation Line bAD-cFS</td>
<td>165</td>
</tr>
<tr>
<td>137</td>
<td>Classify Separation Line bIS-cNA</td>
<td>166</td>
</tr>
<tr>
<td>138</td>
<td>Classify Separation Line cAD-bAD</td>
<td>166</td>
</tr>
<tr>
<td>139</td>
<td>Classify Separation Line cAD-bIS</td>
<td>167</td>
</tr>
<tr>
<td>140</td>
<td>Classify Separation Line cAD-cFS</td>
<td>167</td>
</tr>
<tr>
<td>141</td>
<td>Classify Separation Line cAD-cNA</td>
<td>168</td>
</tr>
<tr>
<td>142</td>
<td>Classify Separation Line cAD-cST</td>
<td>168</td>
</tr>
<tr>
<td>143</td>
<td>Classify Separation Line cAD-dFS</td>
<td>169</td>
</tr>
<tr>
<td>144</td>
<td>Classify Separation Line cFS-bIS</td>
<td>169</td>
</tr>
<tr>
<td>145</td>
<td>Classify Separation Line cFS-cNA</td>
<td>170</td>
</tr>
<tr>
<td>146</td>
<td>Classify Separation Line cFS-cST</td>
<td>170</td>
</tr>
<tr>
<td>147</td>
<td>Classify Separation Line cFS-dFS</td>
<td>171</td>
</tr>
<tr>
<td>148</td>
<td>Classify Separation Line cST-bIS</td>
<td>171</td>
</tr>
<tr>
<td>149</td>
<td>Classify Separation Line cST-cNA</td>
<td>172</td>
</tr>
<tr>
<td>150</td>
<td>Classify Separation Line dFS-bIS</td>
<td>172</td>
</tr>
<tr>
<td>151</td>
<td>Classify Separation Line dFS-cNA</td>
<td>173</td>
</tr>
<tr>
<td>152</td>
<td>Classify Separation Line dFS-cST</td>
<td>173</td>
</tr>
<tr>
<td>153</td>
<td>knn n=1 dist=Cityblock</td>
<td>174</td>
</tr>
<tr>
<td>154</td>
<td>knn n=1 dist=Euclidian</td>
<td>174</td>
</tr>
<tr>
<td>155</td>
<td>knn n=2 dist=Cityblock</td>
<td>174</td>
</tr>
<tr>
<td>156</td>
<td>knn n=2 dist=Euclidian</td>
<td>174</td>
</tr>
<tr>
<td>157</td>
<td>knn n=3 dist=Cityblock</td>
<td>175</td>
</tr>
<tr>
<td>158</td>
<td>knn n=3 dist=Euclidian</td>
<td>175</td>
</tr>
<tr>
<td>159</td>
<td>SVM results</td>
<td>175</td>
</tr>
</tbody>
</table>
**Classify** Function

One against one:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;linear&quot;</td>
<td>&quot;CAD&quot;</td>
<td>bAD</td>
<td>&quot;CF5&quot;</td>
<td>&quot;CF5&quot;</td>
<td>&quot;CT5&quot;</td>
<td>&quot;BIS&quot;</td>
<td>&quot;CNA&quot;</td>
<td>&quot;DNA&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;CAD&quot;</td>
<td>[]</td>
<td>270/400</td>
<td>205/400</td>
<td>310/400</td>
<td>205/400</td>
<td>295/400</td>
<td>799/400</td>
<td>256/400</td>
</tr>
<tr>
<td>3</td>
<td>&quot;CAD&quot;</td>
<td>93/117</td>
<td>[]</td>
<td>101/117</td>
<td>94/117</td>
<td>92/117</td>
<td>89/117</td>
<td>93/117</td>
<td>93/117</td>
</tr>
<tr>
<td>4</td>
<td>&quot;CF5&quot;</td>
<td>295/400</td>
<td>294/400</td>
<td>[]</td>
<td>278/400</td>
<td>271/400</td>
<td>270/400</td>
<td>287/400</td>
<td>275/400</td>
</tr>
<tr>
<td>5</td>
<td>&quot;CF5&quot;</td>
<td>184/226</td>
<td>197/226</td>
<td>190/226</td>
<td>[]</td>
<td>182/226</td>
<td>194/226</td>
<td>185/226</td>
<td>184/226</td>
</tr>
<tr>
<td>6</td>
<td>&quot;CT5&quot;</td>
<td>147/103</td>
<td>144/103</td>
<td>145/103</td>
<td>135/103</td>
<td>[]</td>
<td>139/103</td>
<td>140/103</td>
<td>156/103</td>
</tr>
<tr>
<td>7</td>
<td>&quot;BIS&quot;</td>
<td>40/49</td>
<td>42/49</td>
<td>43/49</td>
<td>39/49</td>
<td>48/49</td>
<td>[]</td>
<td>29/49</td>
<td>47/49</td>
</tr>
<tr>
<td>8</td>
<td>&quot;CNA&quot;</td>
<td>111/186</td>
<td>111/186</td>
<td>111/186</td>
<td>111/186</td>
<td>[]</td>
<td>114/186</td>
<td>[]</td>
<td>[]</td>
</tr>
<tr>
<td>9</td>
<td>&quot;DNA&quot;</td>
<td>150/210</td>
<td>143/210</td>
<td>155/210</td>
<td>168/210</td>
<td>157/210</td>
<td>166/210</td>
<td>159/210</td>
<td>[]</td>
</tr>
</tbody>
</table>

**Figure 126 Classify Function Linear Distance**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;diaginear&quot;</td>
<td>&quot;CAD&quot;</td>
<td>bAD</td>
<td>&quot;CF5&quot;</td>
<td>&quot;CF5&quot;</td>
<td>&quot;CT5&quot;</td>
<td>&quot;BIS&quot;</td>
<td>&quot;CNA&quot;</td>
<td>&quot;DNA&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;CAD&quot;</td>
<td>[]</td>
<td>274/400</td>
<td>293/400</td>
<td>299/400</td>
<td>292/400</td>
<td>296/400</td>
<td>278/400</td>
<td>297/400</td>
</tr>
<tr>
<td>3</td>
<td>&quot;CAD&quot;</td>
<td>93/117</td>
<td>[]</td>
<td>95/117</td>
<td>93/117</td>
<td>87/117</td>
<td>94/117</td>
<td>89/117</td>
<td>89/117</td>
</tr>
<tr>
<td>4</td>
<td>&quot;CF5&quot;</td>
<td>294/400</td>
<td>294/400</td>
<td>[]</td>
<td>279/400</td>
<td>292/400</td>
<td>276/400</td>
<td>279/400</td>
<td>294/400</td>
</tr>
<tr>
<td>5</td>
<td>&quot;CF5&quot;</td>
<td>179/226</td>
<td>189/226</td>
<td>186/226</td>
<td>[]</td>
<td>179/226</td>
<td>182/226</td>
<td>184/226</td>
<td>178/226</td>
</tr>
<tr>
<td>6</td>
<td>&quot;CT5&quot;</td>
<td>130/183</td>
<td>129/183</td>
<td>123/183</td>
<td>126/183</td>
<td>[]</td>
<td>126/183</td>
<td>123/183</td>
<td>135/183</td>
</tr>
<tr>
<td>7</td>
<td>&quot;BIS&quot;</td>
<td>36/49</td>
<td>37/49</td>
<td>37/49</td>
<td>37/49</td>
<td>40/49</td>
<td>[]</td>
<td>40/49</td>
<td>44/49</td>
</tr>
<tr>
<td>8</td>
<td>&quot;CNA&quot;</td>
<td>156/268</td>
<td>156/268</td>
<td>156/268</td>
<td>156/268</td>
<td>[]</td>
<td>156/268</td>
<td>[]</td>
<td>[]</td>
</tr>
<tr>
<td>9</td>
<td>&quot;DNA&quot;</td>
<td>151/210</td>
<td>143/210</td>
<td>155/210</td>
<td>168/210</td>
<td>157/210</td>
<td>166/210</td>
<td>159/210</td>
<td>[]</td>
</tr>
</tbody>
</table>

**Figure 127 Classify Function Diagonal Distance**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;diagquadratic&quot;</td>
<td>&quot;CAD&quot;</td>
<td>bAD</td>
<td>&quot;CF5&quot;</td>
<td>&quot;CF5&quot;</td>
<td>&quot;CT5&quot;</td>
<td>&quot;BIS&quot;</td>
<td>&quot;CNA&quot;</td>
<td>&quot;DNA&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;CAD&quot;</td>
<td>[]</td>
<td>280/400</td>
<td>279/400</td>
<td>294/400</td>
<td>292/400</td>
<td>291/400</td>
<td>290/400</td>
<td>291/400</td>
</tr>
<tr>
<td>3</td>
<td>&quot;CAD&quot;</td>
<td>96/117</td>
<td>[]</td>
<td>95/117</td>
<td>95/117</td>
<td>96/117</td>
<td>96/117</td>
<td>96/117</td>
<td>96/117</td>
</tr>
<tr>
<td>4</td>
<td>&quot;CF5&quot;</td>
<td>266/400</td>
<td>279/400</td>
<td>[]</td>
<td>279/400</td>
<td>276/400</td>
<td>276/400</td>
<td>274/400</td>
<td>276/400</td>
</tr>
<tr>
<td>5</td>
<td>&quot;CF5&quot;</td>
<td>166/226</td>
<td>169/226</td>
<td>175/226</td>
<td>[]</td>
<td>102/226</td>
<td>102/226</td>
<td>104/226</td>
<td>175/226</td>
</tr>
<tr>
<td>6</td>
<td>&quot;CT5&quot;</td>
<td>121/183</td>
<td>114/183</td>
<td>110/183</td>
<td>114/183</td>
<td>[]</td>
<td>117/183</td>
<td>121/183</td>
<td>122/183</td>
</tr>
<tr>
<td>7</td>
<td>&quot;BIS&quot;</td>
<td>42/49</td>
<td>39/49</td>
<td>39/49</td>
<td>42/49</td>
<td>41/49</td>
<td>[]</td>
<td>45/49</td>
<td>44/49</td>
</tr>
<tr>
<td>8</td>
<td>&quot;CNA&quot;</td>
<td>156/268</td>
<td>193/268</td>
<td>158/268</td>
<td>196/268</td>
<td>156/268</td>
<td>156/268</td>
<td>[]</td>
<td>107/158</td>
</tr>
<tr>
<td>9</td>
<td>&quot;DNA&quot;</td>
<td>145/210</td>
<td>141/210</td>
<td>144/210</td>
<td>163/210</td>
<td>146/210</td>
<td>157/210</td>
<td>149/210</td>
<td>[]</td>
</tr>
</tbody>
</table>

**Figure 128 Classify Function Diagonal Quadratic Distance**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;mahalanobis&quot;</td>
<td>&quot;CAD&quot;</td>
<td>bAD</td>
<td>&quot;CF5&quot;</td>
<td>&quot;CF5&quot;</td>
<td>&quot;CT5&quot;</td>
<td>&quot;BIS&quot;</td>
<td>&quot;CNA&quot;</td>
<td>&quot;DNA&quot;</td>
</tr>
<tr>
<td>2</td>
<td>&quot;CAD&quot;</td>
<td>[]</td>
<td>231/400</td>
<td>249/400</td>
<td>281/400</td>
<td>271/400</td>
<td>259/400</td>
<td>249/400</td>
<td>249/400</td>
</tr>
<tr>
<td>3</td>
<td>&quot;CAD&quot;</td>
<td>91/117</td>
<td>[]</td>
<td>97/117</td>
<td>101/117</td>
<td>85/117</td>
<td>93/117</td>
<td>87/117</td>
<td>97/117</td>
</tr>
<tr>
<td>4</td>
<td>&quot;CF5&quot;</td>
<td>273/400</td>
<td>286/400</td>
<td>[]</td>
<td>256/400</td>
<td>274/400</td>
<td>271/400</td>
<td>261/400</td>
<td>272/400</td>
</tr>
<tr>
<td>5</td>
<td>&quot;CF5&quot;</td>
<td>169/226</td>
<td>169/226</td>
<td>[]</td>
<td>165/226</td>
<td>184/226</td>
<td>276/228</td>
<td>152/228</td>
<td>152/228</td>
</tr>
<tr>
<td>6</td>
<td>&quot;CT5&quot;</td>
<td>148/183</td>
<td>135/183</td>
<td>136/183</td>
<td>135/183</td>
<td>[]</td>
<td>139/183</td>
<td>141/183</td>
<td>135/183</td>
</tr>
<tr>
<td>7</td>
<td>&quot;BIS&quot;</td>
<td>40/49</td>
<td>43/49</td>
<td>43/49</td>
<td>45/49</td>
<td>[]</td>
<td>40/49</td>
<td>46/49</td>
<td>46/49</td>
</tr>
<tr>
<td>8</td>
<td>&quot;CNA&quot;</td>
<td>93/158</td>
<td>87/158</td>
<td>90/158</td>
<td>82/158</td>
<td>109/158</td>
<td>92/158</td>
<td>[]</td>
<td>102/158</td>
</tr>
<tr>
<td>9</td>
<td>&quot;DNA&quot;</td>
<td>154/210</td>
<td>136/210</td>
<td>149/210</td>
<td>161/210</td>
<td>137/210</td>
<td>150/210</td>
<td>162/210</td>
<td>[]</td>
</tr>
</tbody>
</table>
Figure 133 Classify Separation Line bAD-cFS

Figure 134 Classify Separation Line bAD-cNA
Figure 137 Classify Separation Line bIS-cNA

Figure 138 Classify Separation Line cAD-bAD
Figure 139 Classify Separation Line cAD-bIS

Figure 140 Classify Separation Line cAD-cFS
Figure 141 Classify Separation Line cAD-cNA

Figure 142 Classify Separation Line cAD-cST
Figure 143 Classify Separation Line cAD-dFS

Figure 144 Classify Separation Line cFS-bIS
Figure 145 Classify Separation Line cFS-cNA

Figure 146 Classify Separation Line cFS-cST

170
Figure 149 Classify Separation Line cST-cNA

Figure 150 Classify Separation Line dFS-bIS
Figure 151 Classify Separation Line dFS-cNA

Figure 152 Classify Separation Line dFS-cST
### knn function

<table>
<thead>
<tr>
<th></th>
<th>OneCityblock</th>
<th>OneEuclidean</th>
<th>TwoCityblock</th>
<th>TwoEuclidean</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>( d' )</td>
<td>( d' )</td>
<td>( d' )</td>
<td>( d' )</td>
</tr>
<tr>
<td>3</td>
<td>( d' )</td>
<td>( d' )</td>
<td>( d' )</td>
<td>( d' )</td>
</tr>
<tr>
<td>5</td>
<td>( s' )</td>
<td>( s' )</td>
<td>( s' )</td>
<td>( s' )</td>
</tr>
<tr>
<td>6</td>
<td>( b' )</td>
<td>( b' )</td>
<td>( b' )</td>
<td>( b' )</td>
</tr>
<tr>
<td>8</td>
<td>( c' )</td>
<td>( c' )</td>
<td>( c' )</td>
<td>( c' )</td>
</tr>
<tr>
<td>9</td>
<td>( h' )</td>
<td>( h' )</td>
<td>( h' )</td>
<td>( h' )</td>
</tr>
</tbody>
</table>

**Figure 153 knn n=1 dist=Cityblock**

<table>
<thead>
<tr>
<th></th>
<th>OneCityblock</th>
<th>OneEuclidean</th>
<th>TwoCityblock</th>
<th>TwoEuclidean</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>( d' )</td>
<td>( d' )</td>
<td>( d' )</td>
<td>( d' )</td>
</tr>
<tr>
<td>3</td>
<td>( d' )</td>
<td>( d' )</td>
<td>( d' )</td>
<td>( d' )</td>
</tr>
<tr>
<td>5</td>
<td>( s' )</td>
<td>( s' )</td>
<td>( s' )</td>
<td>( s' )</td>
</tr>
<tr>
<td>6</td>
<td>( b' )</td>
<td>( b' )</td>
<td>( b' )</td>
<td>( b' )</td>
</tr>
<tr>
<td>8</td>
<td>( c' )</td>
<td>( c' )</td>
<td>( c' )</td>
<td>( c' )</td>
</tr>
<tr>
<td>9</td>
<td>( h' )</td>
<td>( h' )</td>
<td>( h' )</td>
<td>( h' )</td>
</tr>
</tbody>
</table>

**Figure 154 knn n=1 dist=Euclidian**

<table>
<thead>
<tr>
<th></th>
<th>OneCityblock</th>
<th>OneEuclidean</th>
<th>TwoCityblock</th>
<th>TwoEuclidean</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>( d' )</td>
<td>( d' )</td>
<td>( d' )</td>
<td>( d' )</td>
</tr>
<tr>
<td>3</td>
<td>( d' )</td>
<td>( d' )</td>
<td>( d' )</td>
<td>( d' )</td>
</tr>
<tr>
<td>5</td>
<td>( s' )</td>
<td>( s' )</td>
<td>( s' )</td>
<td>( s' )</td>
</tr>
<tr>
<td>6</td>
<td>( b' )</td>
<td>( b' )</td>
<td>( b' )</td>
<td>( b' )</td>
</tr>
<tr>
<td>8</td>
<td>( c' )</td>
<td>( c' )</td>
<td>( c' )</td>
<td>( c' )</td>
</tr>
<tr>
<td>9</td>
<td>( h' )</td>
<td>( h' )</td>
<td>( h' )</td>
<td>( h' )</td>
</tr>
</tbody>
</table>

**Figure 155 knn n=2 dist=Cityblock**

<table>
<thead>
<tr>
<th></th>
<th>OneCityblock</th>
<th>OneEuclidean</th>
<th>TwoCityblock</th>
<th>TwoEuclidean</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>( d' )</td>
<td>( d' )</td>
<td>( d' )</td>
<td>( d' )</td>
</tr>
<tr>
<td>3</td>
<td>( d' )</td>
<td>( d' )</td>
<td>( d' )</td>
<td>( d' )</td>
</tr>
<tr>
<td>5</td>
<td>( s' )</td>
<td>( s' )</td>
<td>( s' )</td>
<td>( s' )</td>
</tr>
<tr>
<td>6</td>
<td>( b' )</td>
<td>( b' )</td>
<td>( b' )</td>
<td>( b' )</td>
</tr>
<tr>
<td>8</td>
<td>( c' )</td>
<td>( c' )</td>
<td>( c' )</td>
<td>( c' )</td>
</tr>
<tr>
<td>9</td>
<td>( h' )</td>
<td>( h' )</td>
<td>( h' )</td>
<td>( h' )</td>
</tr>
</tbody>
</table>

**Figure 156 knn n=2 dist=Euclidian**
### SVM function

#### Figure 157 knn n=3 dist=Cityblock

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>'ThreeCityblock'</td>
<td>'cAD'</td>
<td>bAD</td>
<td>'cFS'</td>
<td>'dFS'</td>
<td>'cST'</td>
<td>'bIS'</td>
<td>cNA</td>
<td>bNA</td>
</tr>
<tr>
<td>2</td>
<td>[ ]</td>
<td>294/400'</td>
<td>271/400'</td>
<td>281/400'</td>
<td>276/400'</td>
<td>205/400'</td>
<td>295/400'</td>
<td>257/400'</td>
</tr>
<tr>
<td>3</td>
<td>92/117'</td>
<td>[ ]</td>
<td>105/117'</td>
<td>95/117'</td>
<td>98/117'</td>
<td>97/117'</td>
<td>98/117'</td>
<td>98/117'</td>
</tr>
<tr>
<td>4</td>
<td>'cFS'</td>
<td>271/400'</td>
<td>279/400'</td>
<td>[ ]</td>
<td>260/400'</td>
<td>291/400'</td>
<td>262/400'</td>
<td>258/400'</td>
</tr>
<tr>
<td>5</td>
<td>'cFS'</td>
<td>191/228'</td>
<td>148/228'</td>
<td>149/228'</td>
<td>[ ]</td>
<td>195/228'</td>
<td>193/228'</td>
<td>156/228'</td>
</tr>
<tr>
<td>6</td>
<td>'cST'</td>
<td>145/103'</td>
<td>148/103'</td>
<td>149/103'</td>
<td>[ ]</td>
<td>146/103'</td>
<td>140/103'</td>
<td>145/103'</td>
</tr>
<tr>
<td>7</td>
<td>'bIS'</td>
<td>43/49'</td>
<td>43/49'</td>
<td>42/49'</td>
<td>40/49'</td>
<td>41/49'</td>
<td>[ ]</td>
<td>45/49'</td>
</tr>
<tr>
<td>8</td>
<td>'cNA'</td>
<td>114/150'</td>
<td>115/150'</td>
<td>116/150'</td>
<td>120/150'</td>
<td>117/150'</td>
<td>117/150'</td>
<td>[ ]</td>
</tr>
<tr>
<td>9</td>
<td>'bNA'</td>
<td>103/210'</td>
<td>169/210'</td>
<td>162/210'</td>
<td>150/210'</td>
<td>165/210'</td>
<td>162/210'</td>
<td>171/210'</td>
</tr>
</tbody>
</table>

#### Figure 158 knn n=3 dist=Euclidian

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>'ThreeEuclidian'</td>
<td>'cAD'</td>
<td>'cAD'</td>
<td>'cFS'</td>
<td>'dFS'</td>
<td>'cST'</td>
<td>'bIS'</td>
<td>cNA</td>
<td>bNA</td>
</tr>
<tr>
<td>2</td>
<td>[ ]</td>
<td>272/400'</td>
<td>275/400'</td>
<td>290/400'</td>
<td>275/400'</td>
<td>264/400'</td>
<td>204/400'</td>
<td>266/400'</td>
</tr>
<tr>
<td>3</td>
<td>95/117'</td>
<td>[ ]</td>
<td>104/117'</td>
<td>92/117'</td>
<td>98/117'</td>
<td>102/117'</td>
<td>98/117'</td>
<td>96/117'</td>
</tr>
<tr>
<td>4</td>
<td>'cFS'</td>
<td>277/400'</td>
<td>278/400'</td>
<td>[ ]</td>
<td>262/400'</td>
<td>278/400'</td>
<td>265/400'</td>
<td>263/400'</td>
</tr>
<tr>
<td>5</td>
<td>'dFS'</td>
<td>192/228'</td>
<td>194/228'</td>
<td>[ ]</td>
<td>194/228'</td>
<td>193/228'</td>
<td>197/228'</td>
<td>202/228'</td>
</tr>
<tr>
<td>6</td>
<td>'cST'</td>
<td>145/103'</td>
<td>146/103'</td>
<td>141/103'</td>
<td>151/103'</td>
<td>[ ]</td>
<td>141/103'</td>
<td>145/103'</td>
</tr>
<tr>
<td>7</td>
<td>'bIS'</td>
<td>49/49'</td>
<td>42/49'</td>
<td>40/49'</td>
<td>39/49'</td>
<td>42/49'</td>
<td>[ ]</td>
<td>42/49'</td>
</tr>
<tr>
<td>8</td>
<td>'cNA'</td>
<td>114/150'</td>
<td>103/150'</td>
<td>122/150'</td>
<td>129/150'</td>
<td>116/150'</td>
<td>119/150'</td>
<td>[ ]</td>
</tr>
<tr>
<td>9</td>
<td>'bNA'</td>
<td>103/210'</td>
<td>169/210'</td>
<td>162/210'</td>
<td>150/210'</td>
<td>165/210'</td>
<td>162/210'</td>
<td>167/210'</td>
</tr>
</tbody>
</table>

#### Figure 159 SVM results

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>'svm'</td>
<td>'cAD'</td>
<td>'bAD'</td>
<td>'cFS'</td>
<td>'dFS'</td>
<td>'cST'</td>
<td>'bIS'</td>
<td>cNA</td>
<td>bNA</td>
</tr>
<tr>
<td>2</td>
<td>[ ]</td>
<td>272/400'</td>
<td>275/400'</td>
<td>295/400'</td>
<td>297/400'</td>
<td>307/400'</td>
<td>260/400'</td>
<td>300/400'</td>
</tr>
<tr>
<td>3</td>
<td>95/117'</td>
<td>[ ]</td>
<td>98/117'</td>
<td>94/117'</td>
<td>97/117'</td>
<td>94/117'</td>
<td>98/117'</td>
<td>96/117'</td>
</tr>
<tr>
<td>4</td>
<td>'cFS'</td>
<td>277/400'</td>
<td>273/400'</td>
<td>[ ]</td>
<td>274/400'</td>
<td>266/400'</td>
<td>272/400'</td>
<td>270/400'</td>
</tr>
<tr>
<td>5</td>
<td>'dFS'</td>
<td>192/228'</td>
<td>185/228'</td>
<td>216/228'</td>
<td>[ ]</td>
<td>194/228'</td>
<td>187/228'</td>
<td>184/228'</td>
</tr>
<tr>
<td>6</td>
<td>'cST'</td>
<td>145/103'</td>
<td>145/103'</td>
<td>139/103'</td>
<td>149/103'</td>
<td>[ ]</td>
<td>150/103'</td>
<td>136/103'</td>
</tr>
<tr>
<td>7</td>
<td>'bIS'</td>
<td>42/49'</td>
<td>37/49'</td>
<td>37/49'</td>
<td>42/49'</td>
<td>44/49'</td>
<td>[ ]</td>
<td>47/49'</td>
</tr>
<tr>
<td>8</td>
<td>'cNA'</td>
<td>99/150'</td>
<td>90/150'</td>
<td>106/150'</td>
<td>104/150'</td>
<td>114/150'</td>
<td>90/150'</td>
<td>[ ]</td>
</tr>
</tbody>
</table>