

UNIVERSITÀ DEGLI STUDI DI PADOVA



**FACOLTÀ DI INGEGNERIA
CORSO DI LAUREA IN BIOINGEGNERIA**

**CLASSIFICATION AND ANALYSIS METHOD
OF LOCAL FIELD POTENTIALS RECORDED
FROM RAT SOMATOSENSORY CORTEX**

DAVIDE TRAVALIN

**SUPERVISOR: PROFESSOR ALESSANDRA BERTOLDO
CO-SUPERVISOR: PROFESSOR STEFANO VASSANELLI
MUFTI MAHMUD**

ACKNOWLEDGEMENTS

I would like to thank Professor Stefano Vassanelli, who gave me the opportunity to work with his research group.

Heartiest thanks to the laboratory staff, dot. Marta Maschietto and Stefano Girardi, who explained me the experiments and provided the signals I have analyzed.

I am grateful to Professor Alessandra Bertoldo of the Department of Information Engineering, who guided me from the mathematical and statistical point of view.

Very special thanks to dot. Mufti Mahmud not only for the help received and the patience whereby he followed me but also (and in this case I want to thank also Mohammed Mostafizur Rahaman) for the opportunity to improve my English and especially for his/their friendship.

INDEX

CONTENTS

List of Figures	Vi
List of Tables	Vii
List of Symbols	Viii
Abstract	1
Chapter 1 Introduction	3
1.1 Introduction to rat whisking system	3
Chapter 2 Background studies	8
2.1 Smoothing	8
2.1.1 Data model	8
2.1.2 Measure error	8
2.1.3 Parameters estimation	9
2.1.4 Residual and weighted residual	12
2.1.5 Estimation error	12
2.1.6 Relative weights	13
2.1.7 Choice of the best model	14
2.2 Pattern recognition	14
2.2.1 Pre-processing	14
2.2.2 Processing	15
2.2.2.1 Contour method	16
2.2.2.2 Matched method	17
2.2.3 Decision	19

2.3 Clustering	20
2.3.1 Introduction to clustering	20
2.3.2 Clustering methods	21
2.3.3 Distances measure	21
2.3.3.1 Euclidean distance	22
2.3.3.2 Manhattan distance	22
2.3.3.3 Minkowski distance	23
2.3.4 Correlation measure	24
2.3.5 Clustering algorithms used	26
2.3.5.1 K-means	26
2.3.5.2 SOM	28
2.3.5.3 Hierarchical clustering	29
2.3.6 Number of clusters in a data set	30
Chapter 3 Materials and Methods	33
3.1 Signal acquisition	33
3.1.1 Signal acquisition setup	33
3.1.2 The experiment	34
3.2 Single sweep analysis	35
3.2.1 Flowchart of the method	35
3.2.1.1 CallerFunction()	35
3.2.1.2 Func_Analysis_Single_Sweep()	36
3.2.2 Algorithm of the method	40
3.2.2.1 Function: func_gen_sgn_big_filt_posS_0 ()	40

3.2.2.2 How to calculate the minimum_pos_matrix vector	41
3.2.2.3 Function: closer_point ()	41
3.2.2.4 How to calculate the first_max_points and ending_points vectors	41
3.2.2.5 Function elimina_punti_specifici()	42
3.2.2.6 Function func_ricerca_fine_risposta()	43
3.2.2.7 How calculate the starting_points vector	43
3.2.2.8 How calculate the single_sweep_recognized vector with the matched filter method	44
3.2.2.9 How calculate the single_sweep_recognized vector with the contour method	46
3.2.2.10 Function func_clustering_kmeans()	47
3.2.2.11 Function func_clustering_som()	49
3.2.2.12 Function func_clustering_agglomerative()	49
3.2.3 Description of the method	51
3.2.3.1 Pre-processing	51
3.2.3.2 Smoothing	53
3.2.3.3 Waveform recognition methods	54
3.2.3.4 Clustering	56
Chapter 4 Results and Discussion	58
4.1 Results and Discussion	58
4.2 Conclusions	66
Bibliografy	67




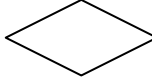



List of Figures

- Figure 1. Wistar rat whiskers
- Figure 2. Whiskers disposition on rat's face
- Figure 3. Barrel cortex pathways
- Figure 4. Noisy measured signal with template superimposed
- Figure 5. Taxicab geometry versus Euclidean distance
- Figure 6. Unit circle shapes corresponding to different values of p
- Figure 7. Examples of one and two dimensional rectangular layouts
- Figure 8. Examples of one step neighborhood of the neuron 3
- Figure 9. Explained variance
- Figure 10. Signal acquisition setup
- Figure 11. Signal recording
- Figure 12. Single sweep versus Average signal
- Figure 13. Single sweep number 100
- Figure 14. Filtered and translated Average signal
- Figure 15. Filtered and translated Single sweep n° 100 with principal point recognized
- Figure 16. Template
- Figure 17. Downsampled template versus Smoothed template
- Figure 18. Template versus Average smoothed template
- Figure 19. Matched filter output 1
- Figure 20. Matched filter output 2
- Figure 21. Contour method output 1
- Figure 22. Contour method output 1
- Figure 23. K-means clustering results

List of Tables

- Table 1. Contingency table for decision of threshold in the event response
- Table 2. Smoothing weighted residuals
- Table 3. Clustering methods results

List of Symbols

Start/Stop Execution	
Calculation / Assignment	
Loop Start / End	
Decision Box	
Function call	
Display	
Connector	

ABSTRACT

A local field potential (LFP) is a particular class of electrophysiological signals, which is related to the sum of all dendritic synaptic activity within a volume of tissue. This type of signal is recorded using a low impedance extracellular microelectrode, placed sufficiently far from individual local neurons to prevent any particular cell from dominating the electrophysiological signal and reflects the sum of action potentials and slower ionic events. The microelectrode measures the electrical potential difference (in volts) between the tissue and a reference electrode placed relatively near the recording electrode.

To extract the LFP from the signal acquired, is used a low-pass filter that removes the spike components allowing only the signal of interest, i.e., field potentials. The LFP represents the accumulative activity of the observed area, whereas, the spike data represents the activity of a particular cell under a given situation. The LFP is composed of the more sustained currents in the tissue, typical of somato-dendritic currents and the major slow current is the postsynaptic potential (PSP).

LFPs are very important to understand the brain activity and network so they are widely studied. Usually, due to the big variability in the individual recordings, it is common to study the average signal obtained from the individual recordings. In this way the average signal obtained is very clean but during this averaging operation many useful information present in the individual recordings are lost. Thus, for a deeper understanding of the neuronal network activity it is necessary to analyze directly the individual LFPs (referred as single sweep).

This work proposes a new method for the single sweep analysis and classification. The method is composed by four principal steps: pre-processing, processing, waveform recognition and clustering.

During the pre-processing the principal points that characterize the single sweep are detected. These points are then used during the processing step to create a single sweep template. To reduce the oscillation inside the template, it is smoothed using well defined smoothing techniques. The next step is the recognition of the defined template in each single sweep (known as template matching). This is done through two techniques: the matched filter method and the contour method.

After the single sweeps recognition, the classification of the detected sweeps is performed. Three different clustering techniques are implemented: the K-means, the SOM (self organizing maps) and the hierarchical agglomerative. Through this method the single sweeps are classified into different binds according to their shape.

Analyzing the various techniques used, we found that the contour method has demonstrated to be the best one in the single sweeps recognition, while the clustering methods proposed generates almost the same results so it is difficult to specify the best one.

To understand the neuronal network activity at a greater level, study of the single sweep LFPs are crucial. Due to the fact that the different shapes in the LFPs represent different network activity, a fine classification method is needed to classify the single sweeps based on their signal shapes. The method proposed in this work performs this classification with a good precision. Though it is difficult to apply many computational operations on a huge dataset as in the case of single sweeps, we tried to use a moderate set of operation that obtained satisfactory results.

CHAPTER 1: INTRODUCTION

1.1 Introduction to rat whisking system

Understanding the brain in terms of its structure, connections, functions and the genetic bases of these properties – remains the central riddle science, especially in neurophysiology. Since the 18th century scientists are trying to decode the mysteries of the brain. To understand the brain and its information processing capability, lots of research has already been done and are still going on. For rodents the whiskers are an important tactile sense organ as the hands are for humans and other primates.



Figure 1. Wistar rat whiskers.

The sensory innervations of each whisker follicle are quite high, reflecting the importance of the information it transmits. In rats, each of the larger follicles receives terminations from approximately 200 trigeminal ganglion cells and the smaller follicles closer to 50. Each muzzle contains about 36 large whiskers, which can be whisked back and forth by muscles in and around the whisker pad, and numerous smaller vibrissae, which do not move and are located around the lip and front of the snout. In total, the muzzle on each side of the face contains approximately 165-210 whiskers depending on the species and strain of animal [1]. The Figure 1 shows a Wister rat.

The barrel cortex is a remarkable structure part of the somatosensory cortex. It receives and processes tactile information derived from the whiskers on the contralateral face of the animal. The barrel cortex is organized in column, each composed of about 10 compartments: L1, L2-Septum-related, L2-Barrel-related, L3-Septum related, L3-Barrel-related, L4-Septum,

L4-Barrel, L5A, L5B, and L6 [1]. In cross-section, the cortex is a six-layered structure where the main input layer is layer IV. The barrels that give the barrel cortex its name are located in layer IV.

Inputs from the thalamus carrying information from a given whisker terminate in discrete areas of layer IV forming anatomically distinguishable areas (barrels) which are separated from each other by septa. And it is found that the “cortical correlates of the mystacial vibrissae” and that “one barrel represents one vibrissa”, [3], (see Figure 2).

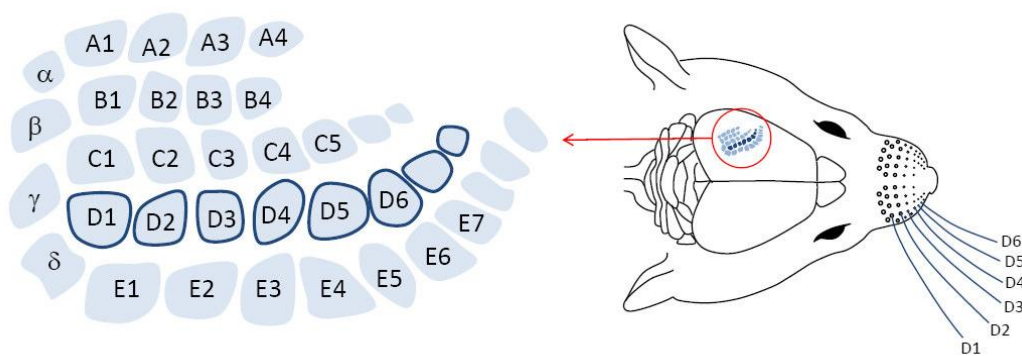


Figure 2. Whiskers disposition on rat's face

Due to its distinctive cytoarchitectonics and functional significance, the barrel cortex has played an important role in neuroscience. The majority of what is known about corticothalamic processing comes from studying the barrel cortex. Researchers have intensively studied the barrel cortex not only at understanding the barrel cortex itself but also at understanding issues in related fields using the barrel cortex as a model system. The barrel cortex has served as a test-bed system for several new methodologies, partly because of its unique and instantly identifiable form, and partly because the species that have barrels, the rodents, are the most commonly used as laboratory mammal.

Neurons within the layer IV barrel cortex directly code for whisker displacement. That is to say, that the neuron within a given barrel will fire when the whisker that barrel represents is moved. These neurons also show directional sensitivity: certain neurons will only fire when the whisker is moved in a specific direction [4], [5]. Neurons in the barrel cortex exhibit the property of synaptic plasticity that allows them to alter the vibrissae to which they respond depending on the rodent's history of tactile experience, [6].

The posteriomedial barrel subfield (PSMBF) is a highly consistent region of the barrel field. Barrels of the PSMBF are the largest and most elliptical in shape and have a striking topographical organization that is identical to that of the major facial whiskers (also called mystacial vibrissae); these whiskers are organized into 5 rows of 4-7 large whiskers that run close to parallel with the bridge of the nose. A consideration of this morphology led Woolsey and Van der Loos to propose that the barrels of the PSMBF were the cortical correlates of the mystacial vibrissae; this is sometimes called the one-barrel-one-vibrissa hypothesis. Thus each barrel in the PSMBF is thought to correspond with a single mystacial vibrissa on the contralateral side of the animal.

Sensory information flows from whisker follicles to somatosensory cortex through at least three parallel pathways: lemniscal, extralemniscal and paralemniscal. The trigeminal nerve carries afferent fibres from the follicles into the brainstem where they synapse with neurons in four different nuclei: principal, interpolar, oral, and caudal. The most important pathway for our purposes is the lemniscal pathway correlated to whiskers deflection. In this pathway, axons from the principal nucleus cross over and project to "barreloids" in the dorsomedial section of the thalamic ventroposterior medial nucleus (VPMdm). Neurons in VPMdm project mainly to barrels in layer IV of primary somatosensory cortex (S1).

The lemniscal input arrives from the ventral posterior medial (VPM) nucleus of thalamus and it has small receptive fields, short response latencies and transmits sensory information about whisker deflection. On the other hand lemniscal input is characterized of large receptive fields and long response latency, and it represents sensory and motor signal as they arise from the collective movements of whiskers during active exploration. Along each column there are some layers, such as L2, L3 and L5B, where there is interaction between the lemniscal and paralemniscal pathway. In other layers, such as L4, the two are processed separately. In the extralemniscal pathway, neurons of the interpolar nucleus project to the ventrolateral section of the VPM (VPMvl). Neurons in VPMvl project to septa between the barrels and to secondary somatosensory cortex (S2). The paralemniscal pathway runs via the interpolar nucleus and the posterior nucleus (POm) of the thalamus to S2 and to diffuse targets in barrel cortex, especially in layer V [7], [8].

A particular sight should be given to layer V: initially this layer was thought to be the paralemniscal cortical relay station, but recently it has been demonstrated that lemniscal and paralemniscal pathways converge on L5. The reason of this characteristic is due to the dendritic architecture of L5A neurons: L5A basal dendrites are not aligned with and do not

respect the barrel borders, moreover L5 apical dendrites bend such that they preferentially pass through the borders of L4 barrels [1]. These different pathways are thought to transmit different modalities of sensory information from the whisker [7], [8] and can be seen in Figure 3.

The well known anatomy of the sensory pathways allows precise electrode placement and the recording of evoked potential (EP) from the vibrissa system. The stimulation of a single vibrissa activates cells in more than one barrel-columns, and, at the same way, neurons from one column respond to more than one whisker with different latency and magnitude. Focusing on the EP, maximal peak-to-peak amplitude can be obtained after stimulation of the corresponding whisker, while EPs evoked by stimulation of whiskers surrounding the principal one have smaller amplitudes [2].

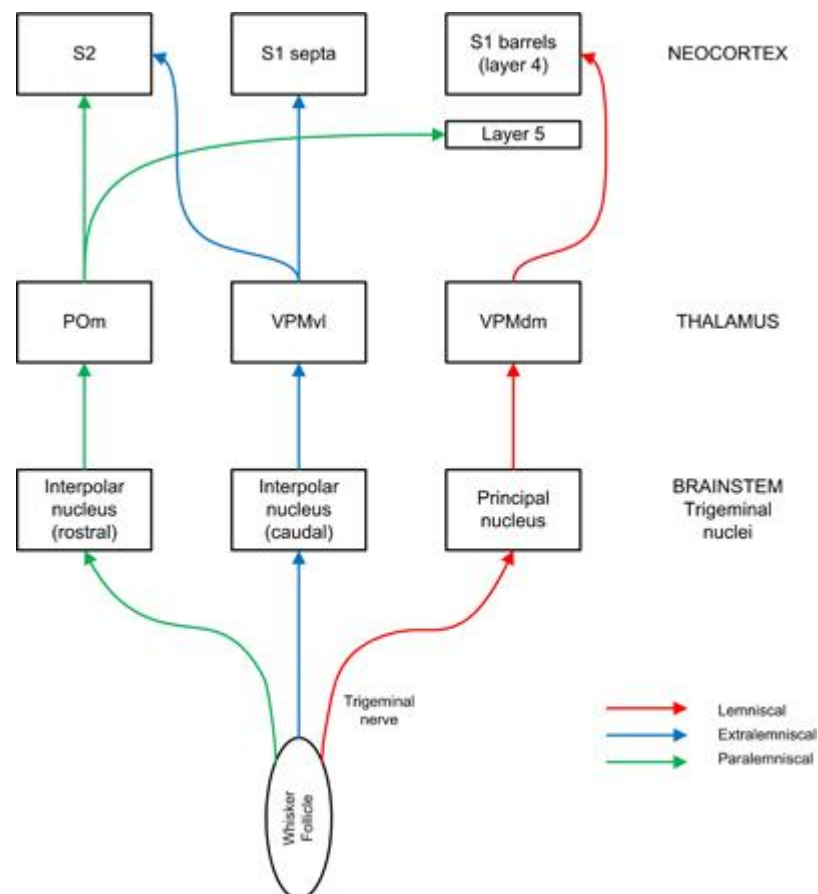


Figure 3. Barrel cortex pathways. Sensory information flows in parallel pathways from whiskers to cortex. (http://en.wikipedia.org/wiki/File:Barrel_cortex_pathways.jpg)

A classical and widely used method for *in vivo* data recording is based on the use of micropipette electrodes inserted into the brain of anesthetized rats. The micropipettes are

used to record potential generated by a population of neurons that is evoked as a response to the applied stimulus. This evoked potential is mainly composed of several action potentials generated by flow of different ionic currents (mainly Na^+ and K^+) controlled by the ion channels present in the neuronal membrane. This extracellular potential is recorded using the micropipette. The use of micropipettes allows to record at different depths and to study signals characterized by different amplitudes.

CHAPTER 2: BACKGROUND STUDIES

2.1 SMOOTHING

In statistics and image processing, to smooth a data set is to create an approximating function that attempts to capture important patterns in the data, while leaving out noise or other fine-scale structures/rapid phenomena.

2.1.1 DATA MODEL

A possible solution to find the best curve to smooth a data set is to define a parametrical data model and to solve the related parametric estimation problem [9]. The predicted model output can be represented as follow:

$$y(t) = g(t, \mathbf{p})$$

Where $\mathbf{p} = [p_1, p_2, \dots, p_M]^T$ is the parameters vector and t is the temporal variable.

The output measured at time sampling t_k is:

$$z_k \quad k = 1, \dots, N$$

Where N is the number of data.

The N measures are affected by a measure error that usually is additive, so the z_k measures are describable as:

$$z_k = y_k + v_k = g(t_k, \mathbf{p}) + v_k$$

Where v_k is the measure error that corrupts the k -th measure z_k .

2.1.2 MEASURE ERROR

Obviously v_k is unknown and it is treated as a random variable. In the most of cases the sequence $\{v_k\}_{k=1\dots N}$ is considered as a random process (usually Gaussian) with zero average (there isn't a systematic error) and independent samples. The variance of the process can be constant (case 1) or not (case 2).

$$\text{case 1: } \forall k, \sigma_k^2 = \sigma^2 \quad ; \quad \text{case 2: } \sigma_k^2 \text{ depends by } k.$$

Often it is convenient to value the measure error entity in relation to data through the variance coefficient CV. This coefficient is given by the ratio between the standard deviation of the measure error (σ_k) and the measure itself (y_k).

$$CV_k = \frac{\sigma_k}{y_k}$$

In many cases it's possible to define a model for the variance, σ_k^2 . The general model has the following expression:

$$\sigma_k^2 = \alpha + \beta \cdot (y_k)^\gamma$$

Some particular cases are:

$$\beta = 0 \rightarrow \sigma_k^2 = \sigma^2 = \alpha \rightarrow \sigma = \sqrt{\alpha} \rightarrow CV_k = \frac{\sqrt{\alpha}}{y_k}$$

$$\alpha = 0, \gamma = 2 \rightarrow \sigma_k^2 = \beta \cdot (y_k)^2 \rightarrow \sigma_k = \sqrt{\beta} \cdot y_k \rightarrow CV_k = CV = \sqrt{\beta}$$

$$\alpha = 0, \gamma = 1 \text{ (Poisson)} \rightarrow \sigma_k^2 = \beta \cdot y_k \rightarrow \sigma_k = \sqrt{\beta \cdot y_k} \rightarrow CV_k = \sqrt{\frac{\beta}{y_k}}$$

2.1.3 PARAMETERS ESTIMATION

The parameters estimation problem consists in the determination of an esteem ($\hat{\mathbf{p}}$) of the parameters vector, knowing the measures (\mathbf{z}) and the output predicted by the model, $g(t, \mathbf{p})$. The measured output, in the generic model, is set to be equal to the sum of the model output and the measure error. The model can be expressed in the following vector form.

$$\mathbf{z} = \mathbf{y} + \mathbf{v} = \mathbf{G}(\mathbf{p}) + \mathbf{v}$$

Where $\mathbf{z} = [z_1, \dots, z_N]^T$, $\mathbf{y} = [y_1, \dots, y_N]^T$, $\mathbf{G}(\mathbf{p}) = [g(t_1, \mathbf{p}), \dots, g(t_N, \mathbf{p})]^T$, \mathbf{v} .

The technique used to calculate the parameters estimation is based on the minimization of the prediction error, $\mathbf{e}(\mathbf{p})$. If true value of \mathbf{p} is unknown, a generic value of \mathbf{p} the vector $\mathbf{e}(\mathbf{p}) = \mathbf{z} - \mathbf{G}(\mathbf{p})$ is used to represent the difference between the data vector and the model prediction for that particular value of \mathbf{p} . It's logical to think that if $\mathbf{e}(\mathbf{p})$ is "small" the value of \mathbf{p} chosen is "good". The goodness of the prediction error vector is evaluated by his norm. The Euclidean norm of a generic vector $\mathbf{x} = [x_1, \dots, x_N]^T$ is given by:

$$\|X\|^2 = \mathbf{x}^T \mathbf{x} = \sum_{i=1}^N x_i^2$$

While the weighted Euclidean norm is given by:

$$\|X\|_{\Phi}^2 = \mathbf{x}^T \Phi \mathbf{x} = \sum_{i=1}^N \frac{x_i^2}{\Phi_i}$$

Where Φ is a square matrix defined positive whose dimensions are $N \times N$.

A basic parameters esteem technique is the least squares estimation, (LS) [9]. Between the all possible values that can be assign to the parameters vector \mathbf{p} , this technique chooses the one that minimize the Euclidean norm of the prediction error vector. The estimation of the parameters vector ($\hat{\mathbf{p}}$) is obtained through the following formula:

$$\hat{\mathbf{p}}_{LS} = \underset{\mathbf{p}}{\operatorname{argmin}} \|\mathbf{e}(\mathbf{p})\|^2 = \underset{\mathbf{p}}{\operatorname{argmin}} [\mathbf{z} - \mathbf{G}(\mathbf{p})]^T [\mathbf{z} - \mathbf{G}(\mathbf{p})]$$

In this case the cost function, $\mathbf{J}(\mathbf{p})$, that is minimized by the least square technique, is equal to the Euclidean norm of $\mathbf{e}(\mathbf{p})$.

Another useful technique to reach the best parameters esteem is the weighted least square estimation, (WLS), [9]. If we consider a matrix W defined positive, it is possible to express the weighted distance between the data and the model through the following quantity:

$$\|\mathbf{z} - \mathbf{G}(\mathbf{p})\|_{W^{-1}}^2 = [\mathbf{z} - \mathbf{G}(\mathbf{p})]^T \mathbf{W}^{-1} [\mathbf{z} - \mathbf{G}(\mathbf{p})]$$

For example, if $W = \operatorname{diag}(w_1, \dots, w_N)$, the data-model distance, weighted through the matrix W , becomes:

$$\|\mathbf{z} - \mathbf{G}(\mathbf{p})\|_{W^{-1}}^2 = \sum_{k=1}^N \frac{e_k^2(\mathbf{p})}{w_k}$$

It is easy to notice that, if the weights are different, the same prediction error, e_k , weights differently in the determination of the distance between data and model. The matrix W can be arbitrary but, if the covariance matrix of the noise, Σ_v , is known, it's possible to demonstrate that the best choice is to consider $W = \Sigma_v$.

So the estimation of the parameters vector calculated through the WLS technique is given by:

$$\hat{\mathbf{p}}_{WLS} = \operatorname{argmin}_{\mathbf{p}} \|\mathbf{e}(\mathbf{p})\|_{\Sigma_v^{-1}}^2 = \operatorname{argmin}_{\mathbf{p}} [\mathbf{z} - \mathbf{G}(\mathbf{p})]^T \Sigma_v^{-1} [\mathbf{z} - \mathbf{G}(\mathbf{p})]$$

In this case the cost function is equal to the weighted Euclidean norm of $\mathbf{e}(\mathbf{p})$.

It is important to notice that the LS and WLS methods produce different estimations only if the measure error variance is not constant. If the model used is a linear parametric model, the measures vector can be expressed as $\mathbf{z} = \mathbf{G}\mathbf{p} + \mathbf{v}$. In this case, \mathbf{G} is a NxM matrix and is no more function of the parameters vector. M is the parameters number of the model.

It is possible to demonstrate that in case of linear parametric model, the WLS expression, used to calculate the parameters estimate, becomes analytically determinable.

$$\hat{\mathbf{p}}_{WLS} = (\mathbf{G}^T \Sigma_v^{-1} \mathbf{G})^{-1} \mathbf{G}^T \Sigma_v^{-1} \mathbf{z}$$

On the contrary, if the model is not linear, the WLS problem has not a closed form solution. In this case some specific iterative methods are required to find the best solution. These techniques can be grouped in two main classes: the gradient methods and direct search methods.

One of the most representative components of the first class is the Gauss-Newton method. The iterative algorithm on which this method is based is presented below.

An initial value is assigned to the parameters vector ($\mathbf{p}^{(k)}$, $k=0$).

Taylor's formula, stopped at the first order, is used to linearize the model around $\mathbf{p}^{(k)}$.

$$g(t_k, \mathbf{p}) = g(t_k, \mathbf{p}^{(k)}) + \left[\frac{\partial g(t_k, \mathbf{p}^{(k)})}{\partial \mathbf{p}_1} \quad \dots \quad \frac{\partial g(t_k, \mathbf{p}^{(k)})}{\partial \mathbf{p}_M} \right] \begin{bmatrix} \mathbf{p}_1 - \mathbf{p}_1^{(k)} \\ \dots \\ \mathbf{p}_M - \mathbf{p}_M^{(k)} \end{bmatrix}$$

Using the precedent formula the problem becomes

$$\begin{bmatrix} z_1 - g(t_1, \mathbf{p}^{(k)}) \\ \dots \\ z_N - g(t_N, \mathbf{p}^{(k)}) \end{bmatrix} = \begin{bmatrix} \left. \frac{\partial g(t_1, \mathbf{p})}{\partial \mathbf{p}_1} \right|_{\mathbf{p}=\mathbf{p}^{(k)}} & \dots & \left. \frac{\partial g(t_1, \mathbf{p})}{\partial \mathbf{p}_M} \right|_{\mathbf{p}=\mathbf{p}^{(k)}} \\ \dots & \dots & \dots \\ \left. \frac{\partial g(t_N, \mathbf{p})}{\partial \mathbf{p}_1} \right|_{\mathbf{p}=\mathbf{p}^{(k)}} & \dots & \left. \frac{\partial g(t_N, \mathbf{p})}{\partial \mathbf{p}_M} \right|_{\mathbf{p}=\mathbf{p}^{(k)}} \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 - \mathbf{p}_1^{(k)} \\ \dots \\ \mathbf{p}_M - \mathbf{p}_M^{(k)} \end{bmatrix} + \begin{bmatrix} v_1 \\ \dots \\ v_N \end{bmatrix}$$

Or in more compact view is given by:

$$\Delta \mathbf{z} = \mathbf{S} \Delta \mathbf{p} + \mathbf{v}$$

Calculus of the esteem of $\Delta \hat{\mathbf{p}}$ through the linear WRS formula.

$$\Delta \hat{\mathbf{p}} = (\mathbf{S}^T \boldsymbol{\Sigma}_v^{-1} \mathbf{S})^{-1} \mathbf{S}^T \boldsymbol{\Sigma}_v^{-1} \mathbf{z}$$

Calculus of the new parameters vector through the following formula:

$$\mathbf{p}^{k+1} = \mathbf{p}^k + \Delta \hat{\mathbf{p}}^k$$

Steps 2-3-4-5 are repeated until the cost function stop falling and/or when:

$$\left| \frac{\Delta \hat{p}_i}{\hat{p}_i} \right| < tol, \quad i = 1, \dots, M$$

2.1.4 RESIDUAL AND WEIGHTED RESIDUAL

Once the estimation of the parameters vector has been detected, the residual vector is defined as the prediction error.

$$\mathbf{r} = \mathbf{z} - \mathbf{G}(\hat{\mathbf{p}})$$

The measure error is:

$$\mathbf{v} = \mathbf{z} - \mathbf{G}(\mathbf{p})$$

Where \mathbf{p} is the real unknown value of the parameters vector, so the residual vector can be reasonably seen as an estimation of the measure error.

$$\mathbf{r} = \mathbf{z} - \mathbf{G}(\hat{\mathbf{p}}) = \hat{\mathbf{v}}$$

If the model is good it's logic to think that the residuals are compatible with the statistic properties of the measure error. Hypothesizing the measure error as a random Gaussian process, with uncorrelated samples and variance matrix known and given by $\boldsymbol{\Sigma}_v = \text{diag}(\sigma_1^2, \dots, \sigma_N^2)$. Considering $\text{var}\left(\frac{v_k}{\sigma_k}\right) = 1$, if we define the weighted residuals as $w\mathbf{r}_k = \frac{r_k}{\sigma_k}$, it is logical to expect that the weighted residuals are uncorrelated and with an amplitude between -1 and +1.

2.1.5 ESTIMATION ERROR

If the estimation of \mathbf{p} depends on the measures and the measures are affected by a measure error, the estimated parameters vector is affected by an estimation error, so how much the

value of \mathbf{p}^\wedge is reliable? The estimation error is defined as $\tilde{\mathbf{p}} = \mathbf{p} - \hat{\mathbf{p}}$ with \mathbf{p}^\wedge an aleatory variable, $\tilde{\mathbf{p}}$ is aleatory too.

The change of the estimation error vector is measured by its covariance matrix that has a form $cov(\tilde{\mathbf{p}}) = E[\tilde{\mathbf{p}}\tilde{\mathbf{p}}^T]$. This matrix gives quantitative information regarding the range that the estimation error can have, so it gives a measure of the precision whereby \mathbf{p} is estimated.

The standard deviation and the confidence interval of the parameters esteem are defined as $SD(\hat{p}_i) = \sqrt{var(\tilde{p}_i)}$ with $\hat{p}_i \pm SD(\hat{p}_i)$. Usually it's preferable to express the estimation precision in relative terms using the variation coefficient.

$$CV(\hat{p}_i) = 100 \frac{SD(\hat{p}_i)}{\hat{p}_i}$$

In case of linear model, we note that:

$$\tilde{\mathbf{p}} = \mathbf{p} - \hat{\mathbf{p}} = [\mathbf{I}_M - (\mathbf{G}^T \boldsymbol{\Sigma}_v^{-1} \mathbf{G})^{-1} \mathbf{G}^T \boldsymbol{\Sigma}_v^{-1} \mathbf{G}] \mathbf{p} - (\mathbf{G}^T \boldsymbol{\Sigma}_v^{-1} \mathbf{G})^{-1} \mathbf{G}^T \boldsymbol{\Sigma}_v^{-1} \mathbf{v}$$

$$cov(\tilde{\mathbf{p}}) = (\mathbf{G}^T \boldsymbol{\Sigma}_v^{-1} \mathbf{G})^{-1} = cov(\hat{\mathbf{p}})$$

On the contrary, in case of non linear model:

$$cov(\tilde{\mathbf{p}}) = cov(\hat{\mathbf{p}}) \cong (\mathbf{S}^T \boldsymbol{\Sigma}_v^{-1} \mathbf{S})^{-1}$$

2.1.6 RELATIVE WEIGHTS

At present measure error covariance matrix has been considered known. It is always possible to define $\boldsymbol{\Sigma}_v = \sigma^2 \mathbf{B}$, so how the parameters estimation problem can be solved if the matrix \mathbf{B} is known while σ^2 is unknown? Considering that:

$$\|\mathbf{z} - \mathbf{G}(\mathbf{p})\|_{\boldsymbol{\Sigma}_v^{-1}}^2 = [\mathbf{z} - \mathbf{G}(\mathbf{p})]^T \boldsymbol{\Sigma}_v^{-1} [\mathbf{z} - \mathbf{G}(\mathbf{p})] = \frac{1}{\sigma^2} [\mathbf{z} - \mathbf{G}(\mathbf{p})]^T \mathbf{B}^{-1} [\mathbf{z} - \mathbf{G}(\mathbf{p})]$$

the solution defined with the "relative weights", $min_p [\mathbf{z} - \mathbf{G}(\mathbf{p})]^T \mathbf{B}^{-1} [\mathbf{z} - \mathbf{G}(\mathbf{p})]$ is equal to the solution that can be found with the "absolute weights" $min_p [\mathbf{z} - \mathbf{G}(\mathbf{p})]^T \boldsymbol{\Sigma}_v^{-1} [\mathbf{z} - \mathbf{G}(\mathbf{p})]$. Therefore the use of the "relative weights" instead of the "absolute weights" doesn't affect the estimation of the parameters vector.

But without σ^2 it is not possible to calculate the precision of the parameters estimation that required the knowledge of the matrix $\boldsymbol{\Sigma}_v$. So a possible solution is given by the calculus of σ^2 after the parameters vector estimation.

$$\hat{\sigma}^2 = \frac{WRSS}{\text{degrees of freedom}} = \frac{[\mathbf{z} - \mathbf{G}(\hat{\mathbf{p}})]^T \mathbf{B}^{-1} [\mathbf{z} - \mathbf{G}(\hat{\mathbf{p}})]}{N - M}$$

2.1.7 CHOICE OF THE BEST MODEL

Increasing the model order, the residuals improve while the precision of the parameters vector estimation worse. To find a good compromise it is possible to use the Akaike criterion or the Schwartz criterion.

$$AIC = WRSS + 2M$$

$$SC = WRSS + M \log N$$

For different model orders the best choice is given by the model corresponding to the lowest value of AIC or SC calculated.

2.2 PATTERN RECOGNITION

Pattern recognition aims to classify data (patterns) based either on *a priori* knowledge or on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations, defining points in an appropriate multidimensional space.

A waveform recognizer can be schematized through a series of three operations, [10], [11]:

1. Pre-processing: it is essentially a pass-band filtering operation that aims to attenuate the unwanted components and to valorize the useful signal.
2. Processing: it is a set of transformations that depends on the recognition method used.
3. Decision: most of the methods used in the elaboration phase generate as output a signal that presents maximum points in correspondence to the waveform occurrence instants. The decision block has to find the maximum point instants through a comparison with a threshold value.

2.2.1 PRE-PROCESSING

The aim of pre-processing is to make the recognition process easier and to increase the signal-to-noise ratio. Considering that the recogniser aims to find the occurrence instants of the waveform and not to evaluate its shape, pre-processing and the next processing step can

introduce modifications on the measured signal. The most immediate pre-processing method is filtering. Supposing to have *a-priori* information about the waveform, it is possible to evaluate its bandwidth and so to apply a band-pass filter. All the energy outside that band is considered noise and therefore filtered. For example, usually there are interferences at 50Hz due to electric lines that can be eliminated through a notch filter, [11].

2.2.2 PROCESSING

Some methods are specific for a determined waveform and they use specific information about the signal to calculate the occurrence instants. These methods are called heuristic methods. Instead other kinds of methods are more general, they suppose to have *a-priori* information about the waveform (template) they want to recognize. Depending on the problem the template changes but the method remains the same. Two kinds of methods part of the class previously presented are the contour method and the matched filter, [11].

Both these methods start with the template (TMP) definition. The template is a waveform which resumes the main characteristic of the considered waveforms. The *a-priori* knowledge about the waveform is used to define the template. Usually TMP is calculated from a training set, a group of N waves that are similar to those that have to be recognized. Eventually these signals could present additive noise. A training set is defined as follow:

$$S_i(k), \quad i = 1, \dots, N, \quad k = 0, \dots, D_i - 1$$

Where N is the number of waves considered and D_i is the duration of the i-th wave.

The template generation required some steps:

1. Alignment of the time instants: the waves are translated in order to align each of them respect to a specific point relative to a structural characteristic well defined. For example in the ECG signal all the training set waves are translated in order to align the vertex R.
2. Duration Adaptation: all the waves considered must have the same length. The easier way to reach this aim is to select the longest signal as reference and to extend all the others aligned signals through a zero-padding operation. Another way is to compress or expand the time axis of each wave with a suitable transform. After this step the new signals are defined as follow:

$$SH_i(k), \quad i = 1, \dots, N, \quad k = 0, \dots, D - 1$$

3. Average operation: the template is obtained through an average operation of the aligned and extended waves.

$$TMP(k) = \frac{1}{N} \sum_{i=1}^N SH_i(k)$$

It is possible to obtain the variance of each TMP samples through the following formula:

$$VTMP(k) = \frac{1}{N} \sum_{i=1}^N [(SH_i(k) - TMP(k))^2]$$

2.2.2.1 CONTOUR METHOD

The contour method uses the template and its variance to generate an upper and a lower bound, [11]. If the samples of the measured signal, including in an interval whose length is equal to the template length, are all comprise between the two bounds then it is recognized the presence of a waveform in the interval. The occurrence instant corresponds to the time position of the first sample included in the interval.

The upper and lower bounds are defined below.

$$Up(k) = TMP(k) + (a \cdot (VTMP(k))^{1/2} + b)$$

$$Low(k) = TMP(k) - (a \cdot (VTMP(k))^{1/2} + b)$$

Where a and b are constants. The way to fix these values will be discussed later.

For each instant n, D values of the measured signal are considered:

$$x(n), x(n - 1), \dots, x(n - D + 1)$$

These samples contain the waveform only if:

$$Low(0) < x(n - D + 1) < Up(0)$$

$$Low(1) < x(n - D + 2) < Up(1)$$

$$Low(D - 1) < \overset{\dots}{x}(n) < Up(D - 1)$$

A visual application of this method is reported below.

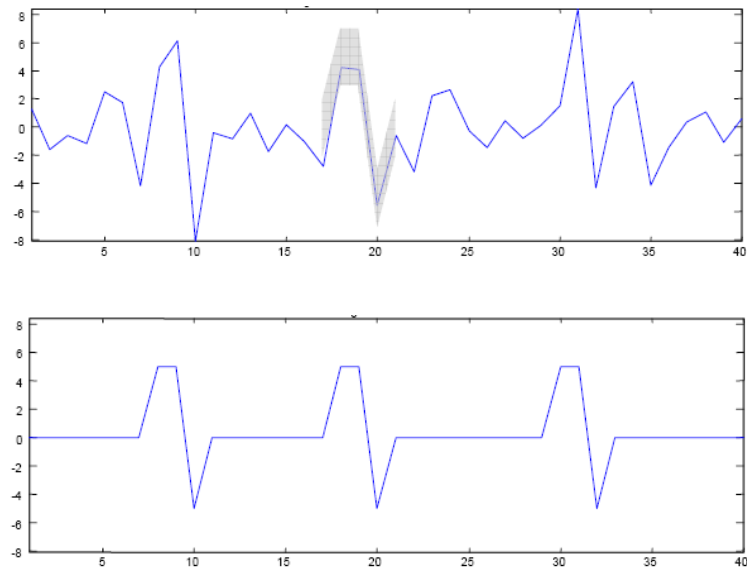


Figure 4. In the first figure is reported the template and its bounds superimposed at the noisy measured signal. It's impossible to recognize directly the presences of the waveforms inside the measured signal. In the second figure is presented the real measured signal not corrupted by the noise.

2.2.2.2 MATCHED FILTER

In telecommunications, a matched filter (originally known as a North filter, [12]) is obtained by correlating a known signal, or template, with an unknown signal to detect the presence of the template in the unknown signal. This is equivalent to convolving the unknown signal with a conjugated time-reversed version of the template (cross-correlation).

The matched filter is the optimal linear filter for maximizing the signal to noise ratio (SNR) in the presence of additive stochastic noise. Matched filters are commonly used in radar, in which a known signal is sent out, and the reflected signal is examined for common elements of the out-going signal. The intuition behind the matched filter relies on correlating the received signal (a vector) with a filter (another vector) that is parallel with the signal, maximizing the inner product.

Inside the generated output signal there are peaks. Each of them can correspond to a waveform occurrence instance but only through an appropriate threshold operation it is possible to select the true peaks that determine the presence of a waveform inside the measured signal. The matched filter consists in a moving average filter (MA) of order M with coefficients b_j ($j=1, \dots, M$). The output signal is generated through the following formula.

$$y(n) = \sum_{j=0}^M b_j x(n-j)$$

Where M is the filter order, b_j is the j -th filter coefficient and $x(n)$ is the input signal.

The value of M is equal to the $D-1$ where D is the length of the template. In this way the output signal, $y(n)$, can contain information about the whole waveform when it is present in the sequence $x(n)$, ..., $x(n-D+1)$. The coefficients of the filter are calculated in order to maximize the difference between the filter output values when in input there is the waveform (case 1) compared to the situation where the input signal doesn't present in any waveform sample (case 2).

Case 1:

$$\begin{aligned} x(n) &= S_i(D-1) + r(n) \\ x(n-1) &= S_i(D-2) + r(n-1) \\ &\dots \\ x(n-M) &= S_i(0) + r(n-M) \end{aligned}$$

Case 2:

$$\begin{aligned} x(n) &= r(n) \\ x(n-1) &= r(n-1) \\ &\dots \\ x(n-M) &= r(n-M) \end{aligned}$$

It is possible to demonstrate that the filter coefficients that satisfy the condition presented above are equal to the samples of the waveforms average, [11]. As esteem of the waveforms average is used the template.

$$\begin{aligned} b_0 &= TMP(M) \\ b_1 &= TMP(M-1) \\ &\dots \\ b_M &= TMP(0) \end{aligned}$$

The filter obtained in this way is known as "matched filter" due to its coefficients are aligned with the waveform samples. The coefficients of the filter, the template and the input signal can be represented by the following vectors.

$$\begin{aligned} \mathbf{b} &= [b_0, b_1, \dots, b_M]^T \\ \mathbf{TMP} &= [TMP(M), TMP(M-1), \dots, TMP(0)]^T \\ \mathbf{x}_n &= [x(n), x(n-1), \dots, x(n-M)]^T \end{aligned}$$

The output represented by the following expression.

$$y(n) = \mathbf{b}^T \mathbf{x}_n = \mathbf{TMP}^T \mathbf{x}_n$$

The last equation establishes that the matched filter output corresponds to the cross-correlation between the observation vector, \mathbf{x}_n , and the **TMP** vector that stores the template samples.

To recognize the waveform occurrence instants the maximum point of the output signal have to be detected through a threshold operation. The waveform recognition process through the matched filter can be resumed by the following steps:

- Through a training-set, the template is generated and stored.
- For each new signal sample, $x(n)$, an observation window, \mathbf{x}_n , long M samples, is built.
- To obtaining the n-th sample of the output signal, $y(n)$, the cross-correlation between **TMP** and \mathbf{x}_n , is calculated.
- $y(n)$ is compared to a threshold: if $y(n)$ is bigger than the threshold value, a waveform is recognized inside the vector \mathbf{x}_n .

2.2.3 DECISION

In both the methods presented it is necessary a decision block. In the contour method it is necessary to fix the best parameters value (a, b) while in the matched filter to find the most correct threshold value. To solve these problem it is necessary a simulated problem. A series of waveforms, whose positions are well known, with additive noise is used to create a simulated problem. Through this signal it is possible to valuate, changing the parameters or the threshold value (depending on the method used), the number of different types of event-response that can occur.

A sample can contain signal and noise or only noise, so the possible events are: (s+r), r.

The possible responses can be: yes (Y), a waveform is recognized, or not (No) there is only noise. So there are four different kinds of event-response, a sample can be classified as:

1. True positive, TP, (s+r)/Y.

2. False positive, FP, r/Y .
3. True negative, TN, r/N .
4. False negative, FN, $(s+r)/N$.

Knowing the waveforms occurrence instants of the build signal, it is possible to create a contingency table.

	Y	N
s+r	TP	FN
r	FP	TN

Table I. Contingency table for decision of threshold in the event response.

Through the values stored in the table, interesting parameters can be calculated.

Sensibility: it is the probability to say yes when there is the waveform. If the threshold decreases this parameter grows, $sensibility = \frac{TP}{(TP+FP)}$.

Specificity: it is the probability to say no when there isn't the waveform. If the threshold decreases this parameter decreases, $specificity = \frac{TN}{(TN+FP)}$.

Correct assignment probability is given by $Pc = \frac{(TP+TN)}{N_{tot}}$; with N_{tot} is the total number of samples.

Usually, the best threshold or parameters values are the ones with maximum last parameter.

2.3 CLUSTERING

2.3.1 INTRODUCTION TO CLUSTERING

The clustering or cluster analysis (term introduced by Robert Tryion in 1939), is a set of techniques of multivariate analysis of data which aim is to select and group elements that are homogeneous inside a data set.

2.3.2 CLUSTERING METHODS

Clustering algorithms may be classified as listed below, [15]:

1. Exclusive Clustering
2. Overlapping Clustering
3. Hierarchical Clustering
4. Probabilistic Clustering

In the first case data are grouped in an exclusive way, so that if a certain datum belongs to a definite cluster then it could not be included in another cluster. On the contrary the second type, the overlapping clustering, uses fuzzy sets to cluster data, so that each point may belong to two or more clusters with different degrees of membership. In this case, data will be associated to an appropriate membership value. Hierarchical clustering algorithm find successive clusters using previously established clusters. These algorithms can be either agglomerative ("bottom-up") or divisive ("top-down"). The first type starts considering each element as a cluster, then, at each step, the algorithm merges the two nearest clusters. The algorithm stops when a certain number of clusters are reached or when the minimum distance between the clusters exceed a particular value. The second one starts considering all the elements part of only one cluster, then the algorithm proceeds, step by step, dividing the initial cluster in an increasing number of dimensionally smaller clusters. The criterion that governs the divisions aims to obtain homogeneous data sets. The algorithm proceeds until is achieved the target number of clusters. Finally, the last kind of clustering uses a completely probabilistic approach.

2.3.3 DISTANCE MEASURES

An important step in any clustering is to select a distance measure, which will determine how the similarity of two elements is calculated. This will influence the shape of the clusters, as some elements may be close to one another according to one distance and farther away according to another.

For example, in a 2-dimensional space, the distance between the point $(x = 1, y = 0)$ and the origin $(x = 0, y = 0)$ is always 1 according to the usual norms, but the distance between the point $(x = 1, y = 1)$ and the origin can be 2, $\sqrt{2}$ or 1 if you take respectively the 1-norm, 2-norm or infinity-norm distance.

Common distance functions are the following.

2.3.3.1 EUCLIDIAN DISTANCE

The Euclidian distance or 2-norm distance is the "ordinary" distance between two points that one would measure with a ruler, and is given by the Pythagorean formula, [13]. By using this formula as distance, Euclidean space (or even any inner product space) becomes a metric space. The associated norm is called the Euclidian norm. The Euclidean distance between points \mathbf{p} and \mathbf{q} is the length of the line segment $\overline{\mathbf{pq}}$.

In Cartesian coordinates, if $\mathbf{p} = (p_1, p_2, \dots, p_n)$ and $\mathbf{q} = (q_1, q_2, \dots, q_n)$ are two points in Euclidean n-space, then the distance from \mathbf{p} to \mathbf{q} is given by:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

The Euclidean norm measures the distance of a point to the origin of Euclidean space:

$$\|\mathbf{p}\| = \sqrt{p_1^2 + p_2^2 + \dots + p_n^2} = \sqrt{\mathbf{p} \cdot \mathbf{p}}$$

Where the last equation involves the dot product. This is the length of \mathbf{p} , when regarded as a Euclidian vector from the origin. The distance itself is given by

$$\|\mathbf{p} - \mathbf{q}\| = \sqrt{(\mathbf{p} - \mathbf{q}) \cdot (\mathbf{p} - \mathbf{q})} = \sqrt{\|\mathbf{p}\|^2 + \|\mathbf{q}\|^2 - 2\mathbf{p} \cdot \mathbf{q}}$$

2.3.3.2 MANHATTAN DISTANCE

The taxicab distance, between two vectors \mathbf{p} and \mathbf{q} in an n-dimensional real vector space with fixed Cartesian coordinate system, is the sum of the lengths of the projections of the line segment between the points onto the coordinate axes, [14]. More formally,

$$d(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^n |p_i - q_i|$$

Where $\mathbf{p} = (p_1, p_2, \dots, p_n)$ and $\mathbf{q} = (q_1, q_2, \dots, q_n)$ are vectors.

For example, in the plane, the taxicab distance between (p_1, p_2) and (q_1, q_2) is $|p_1 - q_1| + |p_2 - q_2|$.

In the following figure are shown the differences between Manhattan and Euclidean distance. The red, blue, and yellow lines have the same length (12) in both Euclidean and taxicab geometry. In Euclidean geometry, the green line has length $6 \times \sqrt{2} \approx 8.48$, and is the unique shortest path. In taxicab geometry, the green line's length is still 12, making it no shorter than any other path shown.

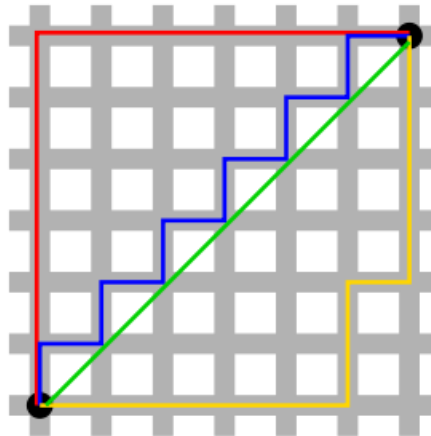


Figure 5. Taxicab geometry versus Euclidean distance.

2.3.3.3 MINKOWSKI DISTANCE

The Minkowski distance is a metric on Euclidean space which can be considered as a generalization of both the Euclidean distance and the Manhattan distance. The distance of order p between two points $\mathbf{p} = (p_1, p_2, \dots, p_n)$ and $\mathbf{q} = (q_1, q_2, \dots, q_n)$ is given by:

$$d(\mathbf{p}, \mathbf{q}) = \left[\sum_{i=1}^n |p_i - q_i|^p \right]^{1/p}$$

Minkowski distance is typically used with p being 1 or 2. The latter is the Euclidean distance, while the former is sometimes known as the Manhattan distance. In the limiting case of p reaching infinity we obtain the Chebyshev distance:

$$\lim_{p \rightarrow \infty} \left(\sum_{i=1}^n |p_i - q_i|^p \right)^{1/p} = \max_{1 \leq i \leq n} |p_i - q_i|$$

In the following figure is shown how the unit circle shape changes with various values of p .



Figure 6. Unit circle shapes corresponding to different values of p .

In general if the components of the data instance vectors are all in the same physical units then it is possible that the simple Euclidean distance metric is sufficient to successfully group similar data instances. However, even in this case the Euclidean distance can sometimes be misleading. It is often the case that the components of the data feature vectors are not immediately comparable. It can be that the components are not continuous variables, like length, but nominal categories, such as the days of the week. In these cases, domain knowledge must be used to formulate an appropriate measure [15].

2.3.4 CORRELATION MEASURE

Another way to determine if two elements can be part of the same cluster consists in the evaluation of their correlation instead of their distance. A type of correlation often used in cluster analysis is the Pearson's correlation, [16]. In statistics, the Pearson product-moment correlation coefficient (sometimes referred to as the PMCC, and typically denoted by r) is a measure of the correlation (linear dependence) between two variables X and Y , giving a value between +1 and -1 inclusive. It is widely used in the sciences as a measure of the strength of linear dependence between two variables. It was developed by Karl Pearson from a similar but slightly different idea introduced by Francis Galton in the 1880s. The correlation coefficient is sometimes called "Pearson's r ". Pearson's correlation coefficient between two variables is defined as the covariance of the two variables divided by the product of their standard deviations:

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

The above formula defines the population correlation coefficient, commonly represented by the Greek letter ρ . Substituting estimates of the covariances and variances based on a sample gives the *sample correlation coefficient*, commonly denoted r :

$$r_{X,Y} = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}}$$

An equivalent expression gives the correlation coefficient as the mean of the products of the standard scores. Based on a sample of paired data (X_i, Y_i) , the sample Pearson correlation coefficient is:

$$r_{X,Y} = \frac{1}{n-1} \sum_{i=1}^n \left(\frac{X_i - \bar{X}}{s_X} \right) \left(\frac{Y_i - \bar{Y}}{s_Y} \right)$$

where $\left(\frac{X_i - \bar{X}}{s_X} \right)$, \bar{X} and s_X are the standard score, the sample, and the sample standard deviation.

The absolute value of both the sample and population Pearson correlation coefficients are less than or equal to 1. Correlations equal to 1 or -1 correspond to data points lying exactly on a line (in the case of the sample correlation), or to a bivariate distribution entirely supported on a line (in the case of the population correlation). The Pearson correlation coefficient is symmetric: $\text{corr}(X,Y) = \text{corr}(Y,X)$.

A key mathematical property of the Pearson correlation coefficient is that it is invariant to changes in location and scale. That is, we may transform X to $a + bX$ and transform Y to $c + dY$, where a , b , c , and d are constants, without changing the correlation coefficient (this fact holds for both the population and sample Pearson correlation coefficients).

The Pearson correlation can be expressed in terms of uncentered moments. Since $\mu_X = E(X)$, $\sigma_X^2 = E[(X - E(X))^2] = E(X^2) - E^2(X)$ and likewise for Y , and since

$$E[(X - E(X))(Y - E(Y))] = E(XY) - E(X)E(Y)$$

we may also write

$$\rho_{X,Y} = \frac{E(XY) - E(X)E(Y)}{\sqrt{E(X^2) - (E(X))^2} \sqrt{E(Y^2) - (E(Y))^2}}$$

Alternative formulae for the sample Pearson correlation coefficient are also available:

$$r_{X,Y} = \frac{\sum_{i=1}^n X_i Y_i - n\bar{X}\bar{Y}}{(n-1)s_X s_Y} = \frac{n \sum X_i Y_i - \sum X_i \sum Y_i}{\sqrt{n \sum X_i^2 - (\sum X_i)^2} \sqrt{n \sum Y_i^2 - (\sum Y_i)^2}}$$

The above formula conveniently suggests a single-pass algorithm for calculating sample correlations, but, depending on the numbers involved, it can sometimes be numerically unstable. Another expression used to define the uncentered correlation is the following:

$$r_{X,Y} = \frac{1}{n} \sum_{i=1}^n \frac{X_i}{\sqrt{\frac{1}{n} \sum_{i=1}^n X_i^2}} \cdot \frac{Y_i}{\sqrt{\frac{1}{n} \sum_{i=1}^n Y_i^2}}$$

The correlation coefficient ranges from -1 to 1 . A value of 1 implies that a linear equation describes the relationship between X and Y perfectly, with all data points lying on a line for which Y increases as X increases. A value of -1 implies that all data points lie on a line for which Y decreases as X increases. A value of 0 implies that there is no linear correlation between the variables.

More generally, note that $(X_i - \bar{X})(Y_i - \bar{Y})$ is positive if and only if X_i and Y_i lie on the same side of their respective means. Thus the correlation coefficient is positive if X_i and Y_i tend to be simultaneously greater than, or simultaneously less than, their respective means. The correlation coefficient is negative if X_i and Y_i tend to lie on opposite sides of their respective means.

2.3.5 CLUSTERING ALGORITHMS USED

In this chapter we propose three of the most used clustering algorithms. The program implemented for single sweep analysis used each of them.

The algorithms are:

- K-means,
- Hierarchical clustering,
- Self organizing maps (SOM).

2.3.5.1 K-MEANS

K-means (MacQueen, 1967, [17]) is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori.

The main idea is to define k centroids, one for each cluster. These centroids should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is

pending, the first step is completed and an early groupage is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more.

Finally, this algorithm aims at minimizing an objective function. For this type of clustering algorithm we propose three different type of objective function that the program user can choose giving a specific data input, [18], [19].

The first objective function aims to minimize the mean of the summation of the Euclidean distances between the centroid of each cluster and the elements that are part of that cluster.

$$J = \frac{1}{N} \sum_{r=1}^K \sqrt{\sum_{x_i \in c_r} (x_i^{(r)} - c_r)^2}$$

Where N is the total number of elements, K is the number of cluster chosen by the program user, c_r is the r-th cluster center and $x_i^{(r)}$ is the i-th element part of the r-th cluster.

The second objective function aims to minimize the summation of the variances of all K cluster.

$$J = \sum_{r=1}^K s_r^2$$

$$\bar{m}_r = \frac{1}{n_r} \sum_{x_g \in c_r} \bar{x}_g$$

$$s_r^2 = \frac{\sum_{x_g \in c_r} \|\bar{x}_g - \bar{m}_r\|^2}{n_r - 1}$$

Where J is the summation of the variance in the K clusters, m_r is the mean of the cluster c_r while s_r^2 is the variance of the cluster c_r .

The last objective function proposed aims to minimize the ratio between the variance inside cluster and among clusters.

$$J = \frac{VAR_{entro}}{VAR_{tra}} = \frac{\frac{\sum_{r=1}^K s_r^2}{K}}{\frac{\sum_{r=1}^K n_r \|\bar{m}_{tot} - \bar{m}_r\|^2}{\sum_{r=1}^K n_r}}$$

Where m_r is the mean of the cluster c_r , s_r^2 is the variance of the cluster c_r and m_{tot} is the mean of the clusters centroids.

2.3.5.2 SOM

A self-organizing map (SOM) or self-organizing feature map (SOFM) is a type of artificial neural networks that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map.

Like most artificial neural networks, SOMs operate in two modes: training and mapping.

Training builds the map using input examples. Mapping automatically classifies a new input vector. A self-organizing map consists of components called nodes or neurons. Associated with each node is a weight vector of the same dimension as the input data vectors and a position in the map space. The usual arrangement of nodes is a regular spacing in a hexagonal or rectangular grid. The self-organizing map describes a mapping from a higher dimensional input space to a lower dimensional map space. The procedure for placing a vector from data space onto the map is to find the node with the closest weight vector to the vector taken from data space and to assign the map coordinates of this node to our vector.

It has been shown that while self-organizing maps with a large number of nodes rearrange data in a way that is fundamentally topological in character, small self-organizing maps behave in a way that is similar to K-means. SOMs work somewhat like K-means clustering but are a little richer. With K-Means, you choose the number of clusters to fit the data into. For a SOM you choose the shape and size of a network of clusters to fit the data into.

The training utilizes competitive learning. When a training example is fed to the network, its Euclidean distance to all weight vectors is computed. The neuron with weight vector most similar to the input is called the best matching unit (BMU). The weights of the BMU and neurons close to it in the SOM lattice are adjusted towards the input vector. The magnitude of the change decreases with time and with distance from the BMU. The update formula for a neuron with weight vector $\mathbf{Wv}(t)$ is the following:

$$\mathbf{Wv}(t + 1) = \mathbf{Wv}(t) + \theta(v,t) \cdot \alpha(t) \cdot (\mathbf{D}(t) - \mathbf{Wv}(t))$$

Where $\alpha(t)$ is a monotonically decreasing learning coefficient, $\mathbf{D}(t)$ is the input vector and $\theta(v,t)$ is the neighborhood function. This function depends on the lattice distance between the BMU and neuron v .

In the simplest form it is one for all neurons close enough to BMU and zero for others, but a gaussian function is a common choice, too. Regardless of the functional form, the neighborhood function shrinks with time [21]. At the beginning when the neighborhood is broad, the self-organizing takes place on the global scale. When the neighborhood has shrunk to just a couple of neurons the weights are converging to local estimates.

This process is repeated for each input vector for a (usually large) number of cycles. The network winds up associating output nodes with groups or patterns in the input data set. During mapping, there will be one single *winning* neuron: the neuron whose weight vector lies closest to the input vector. This can be simply determined by calculating the Euclidean distance between input vector and weight vector.

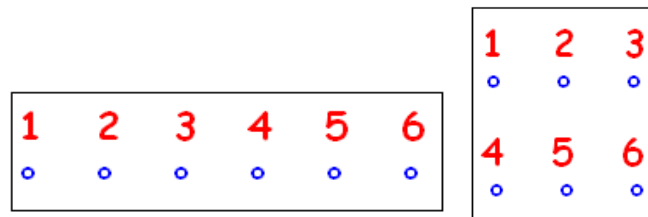


Figure 7. Examples of one and two dimensional rectangular layouts, [19]

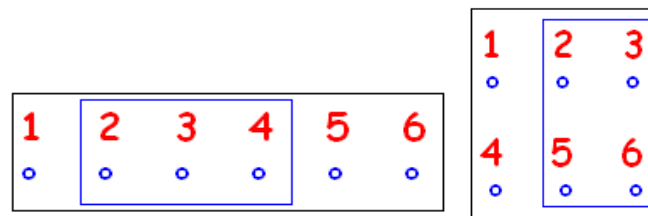


Figure 8. Examples of one step neighborhood of the neuron 3, [19]

2.3.5.3 HIERARCHICAL CLUSTERING

Algorithms for hierarchical clustering are generally either agglomerative, in which one starts at the leaves and successively merges clusters together; or divisive, in which one starts at the root and recursively splits the clusters. Any valid metric may be used as a measure of similarity between pairs of observations. The choice of which clusters to merge or split is

determined by a linkage criterion, which is a function of the pairwise distances between observations, [19], [22].

In *single-linkage* clustering (also called the *connectedness* or *minimum* method) we consider the distance between one cluster and another cluster to be equal to the shortest distance from any member of one cluster to any member of the other cluster. If the data consist of similarities, we consider the similarity between one cluster and another cluster to be equal to the greatest similarity from any member of one cluster to any member of the other cluster.

In *complete-linkage* clustering (also called the *diameter* or *maximum* method) we consider the distance between one cluster and another cluster to be equal to the greatest distance from any member of one cluster to any member of the other cluster.

In *average-linkage* clustering, we consider the distance between one cluster and another cluster to be equal to the average distance from any member of one cluster to any member of the other cluster.

So if C_{new} is the new cluster just created and X is a generic element, the three methods presented above to update the distances matrix are defined as follow:

Nearest Neighbor or Single Linkage (SL):

$$d(X, C_{new}) = \min\{d(\forall x \in X, \forall y \in C_{new})\}$$

Farthest Neighbor or Complete Linkage (CL):

$$d(X, C_{new}) = \max\{d(\forall x \in X, \forall y \in C_{new})\}$$

Average Linkage (AL) or Unweighted Pair Group Method Average (UPGMA):

$$d(X, C_{new}) = \text{mean}\{d(\forall x \in X, \forall y \in C_{new})\}$$

2.3.6 NUMBER OF CLUSTERS IN A DATA SET

The number of cluster in a data set is a quantity often labeled K , is fundamental to the problem of data clustering, and is a distinct issue from the process of actually solving the clustering problem. In most cases, K must be chosen somehow and specified as an input parameter to clustering algorithms, with the exception of methods such as correlation

clustering, which are able to determine the optimal number of clusters during the course of the algorithm.

The correct choice of K is often ambiguous, with interpretations depending on the shape and scale of the distribution of points in a data set and the desired clustering resolution of the user. In addition, increasing K without penalty will always reduce the amount of error in the resulting clustering, to the extreme case of zero error if each data point is considered its own cluster (i.e., when K equals the number of data points, N). Intuitively then, the optimal choice of K will strike a balance between maximum compression of the data using a single cluster, and maximum accuracy by assigning each data point to its own cluster.

If an appropriate value of K is not apparent from a-prior knowledge of the properties of the data set, it must be chosen somehow. There are several categories of methods for making this decision. One simple rule of thumb sets the number to:

$$K \approx \left(\frac{N}{2}\right)^{1/2}$$

Where N is the number of objects (data points), [23].

Another method looks at the percentage of variance explained as a function of the number of clusters. You should choose a number of clusters so that adding another cluster doesn't give much better modeling of the data. More precisely, if you graph the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information (explain a lot of variance), but at some point the marginal gain will drop, giving an angle in the graph. The clusters numbers are chosen at this point, hence the "elbow criterion". This "elbow" cannot always be unambiguously identified [24].

Percentage of variance explained is the ratio of the between-group variance to the total variance. A slight variation of this method plots the curvature of the within group variance [25].

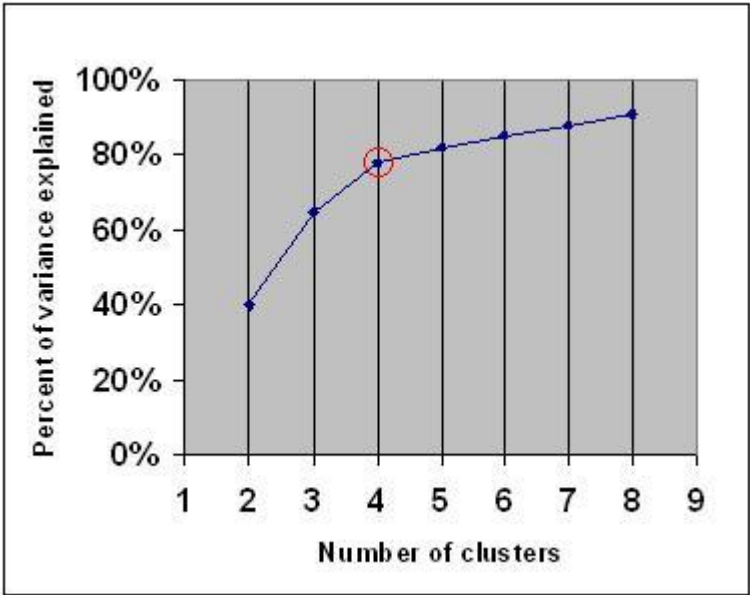


Figure 9. Explained variance, the “elbow” is indicated by the red circle. The number of clusters chosen should be four.

CHAPTER 3: MATERIALS AND METHODS

3.1 SIGNAL ACQUISITION

3.1.1 SIGNAL ACQUISITION SETUP

The signals analyzed in this work are obtained from the somatosensory cortex (S1) of the rat using micropipettes. The rats used are P30-P40 male Wistar rats (Charles River International Inc.). The electrophysiological experiment is realized using borosilicate pipette (0.6-1 M Ω resistance), filled with a standard Krebs's solution (in mM: NaCl 120, KCl 1.99, NaHCO₃ 25.56, KH₂PO₄ 136.09, CaCl₂ 2, MgSO₄ 1.2, glucose 11).

The Figure 10 shows the recording setup. The micropipette is fixed to a micromanipulator at 45° tilted to the vertical axis of the manipulator, thus being inserted perpendicularly to S1 cortex surface. The micromanipulator is used to reach the brain area of interest as it allows precise movements along the three axes. A chlorinated silver electrode, placed within the pipette, is used as recording electrode. Two additional chlorinated electrodes are placed in the bath solution and are both used as reference electrodes due to the differential recording mode of the amplifier. The electrodes are connected to a pre-amplifier (SEC-EXT) which in turn is connected to the amplifier (SEC-10L, npi electronic GmbH) used for recording.

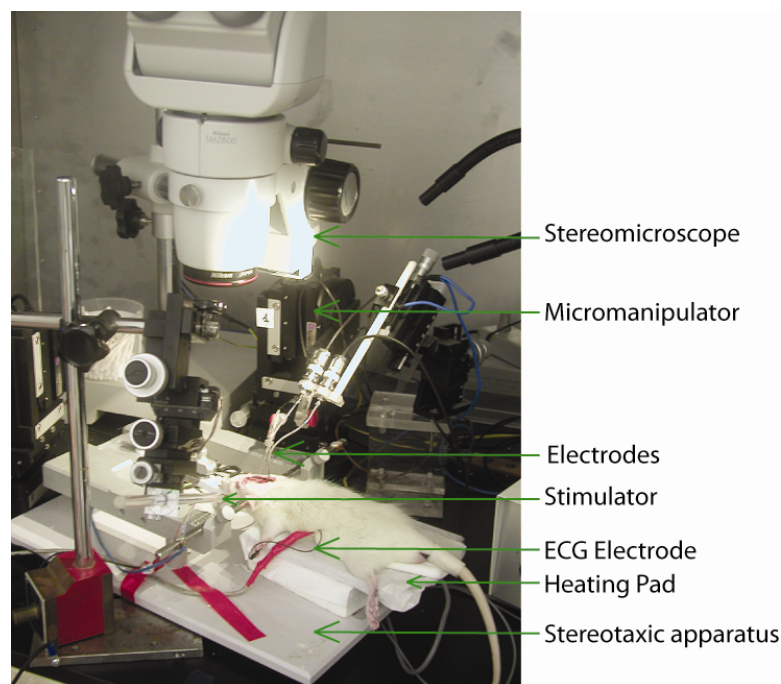


Figure 10. Signal acquisition setup.

The mechanical stimulation is made by a custom-made speaker which allows whisker movements through a small capillary where the selected whisker is inserted. The stimulus is obtained by applying a square pulse, with 2 V amplitude and 50 ms duration, generated by the AGILENT waveform generator (Agilent 33250A 80 MHz, Agilent Technologies).

Signals are acquired by computer using the open source software “WinWCP” (Version: 4.1.0) developed by the SIPBS, University of Strathclyde, UK (http://spider.science.strath.ac.uk/sipbs/software_ses.htm).

3.1.2 THE EXPERIMENT

To record signals from the brain, neurosurgery is done to open the skull of the rat and to expose the brain area of interest. Rats are anesthetized with a mixture of two different anesthetics: Xylazine (1.4 g/100 g weight) for inducing muscle relaxation, and Tiletamine that works as a painkiller (2 mg/100 g weight). The anesthesia level is monitored throughout the experiment by testing eye and hind-limb reflexes, respiration and checking the absence of whiskers spontaneous movements. Whenever necessary, additional doses of Tiletamine (0.5 mg/100 g weight) and Xylazine (0.5 mg/100 g weight) are provided.

During both the surgery and recording section, animals are positioned on a common stereotaxic apparatus under a stereomicroscope. The head of the rat is fixed by means of two ear-bars and a teeth-bar. While the rats are anesthetized, other parameters are constantly monitored: the heart beat is controlled by ECG recorded using a Bio-amplifier (DAM, WPI instr.) with two measuring electrodes placed on the forelimbs and the ground electrode on a hindlimb. The body temperature is checked with a temperature controller (ATC 1000, WPI instr.) that uses a combination of a heating pad and an anal probe to establish a feedback loop to keep the rat warm depending on the actual body temperature measured by the probe. The neurosurgery is performed to expose the target area of the brain for recording. The head's skin is cut starting from the imaginary line that connects the eyes until the imaginary line that connects the ears. To remove the connective tissue between the skin and the skull, a bone scraper is used.

The next step before reaching the brain is the skull opening in correspondence of the right S1 somatosensory cortex and the removal of the meninges. The stereomicroscope is used to recognize precisely the S1 area by means of a set of coordinates in the ocular lens. These coordinates are referred to the bregma (anatomical point of the skull at which the coronal suture is intersected perpendicularly by the sagittal suture).

During the surgery and recording session, the brain is bathed by the standard Krebs's solution, constantly oxygenated and warmed at 37 °C.

Each whisker, trimmed at about 10 mm from the mystacial pad, is individually inserted into the speaker tube, and the corresponding response is checked at 750 µm depth (layer IV) in order to find the most responsive whisker for the selected recording point in the cortex. The "principal whisker" is then chosen for the recording and evoked LFPs are recorded from all cortical layers with 90 µm pitch.

In each experiment section, from the total S1 depth, neuronal signals are recorded with a 20 kHz sample rate. For each depth, 100 sweeps of 500 ms duration are recorded, with 500 ms delay between each sweep and with a 150 ms delay before the stimulus. Figure 11 shows the rat during a signal acquisition experiment.

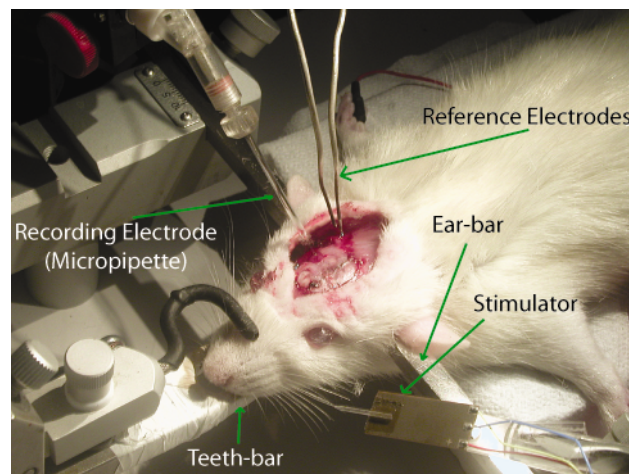


Figure 11. Signal recording from the S1 cortex using a micropipette with reference electrodes in the bath.

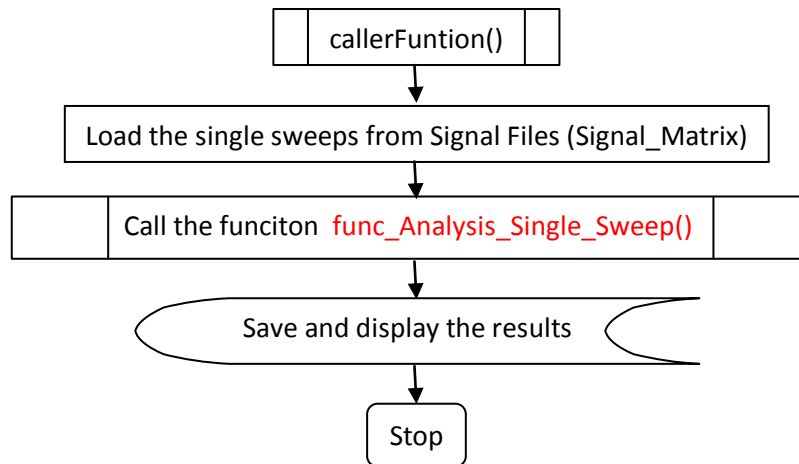
3.2 SINGLE SWEEP ANALYSIS

3.2.1 FLOWCHART OF THE METHOD

(The words written in **bold** characters are variable names used inside the program.)

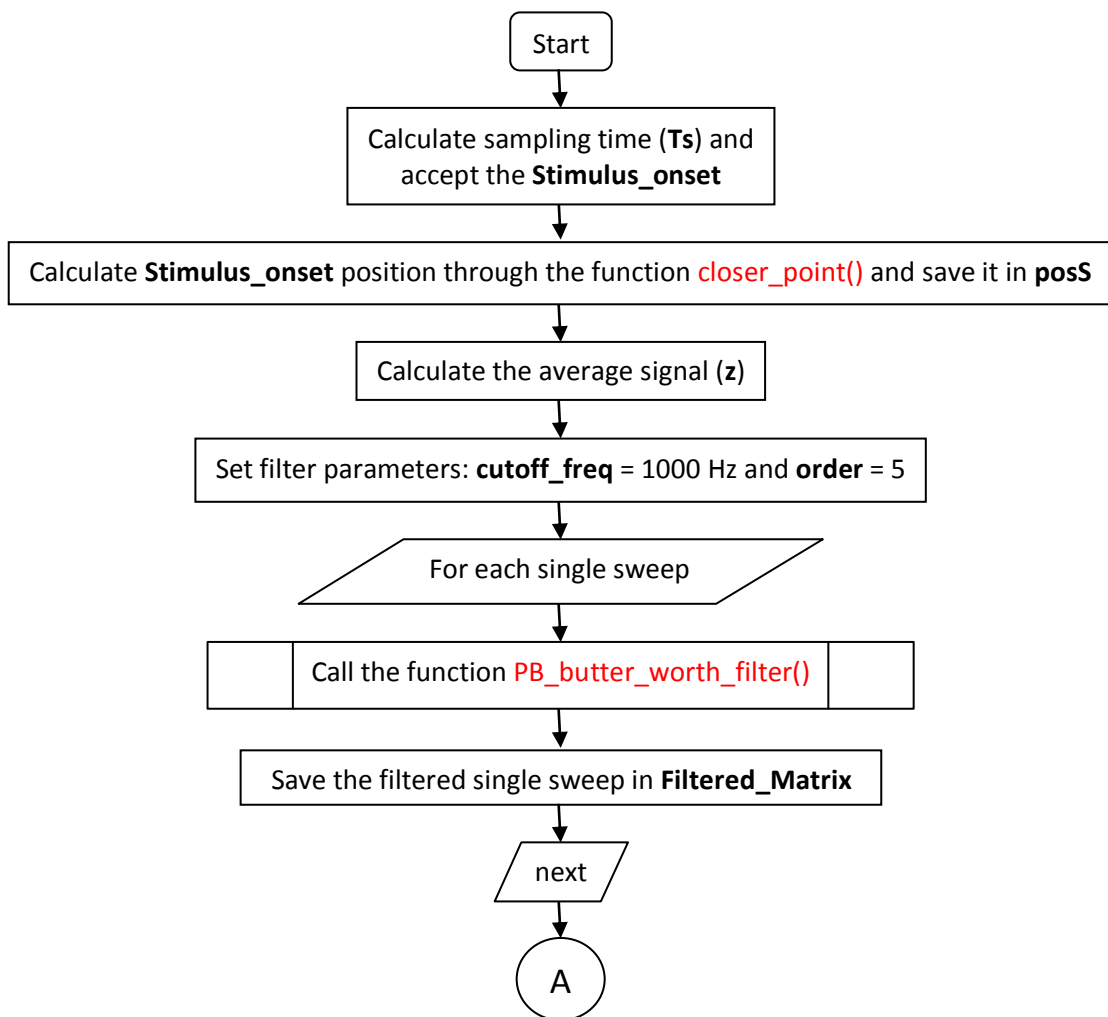
3.2.1.1 *CALLERFUNCTION*

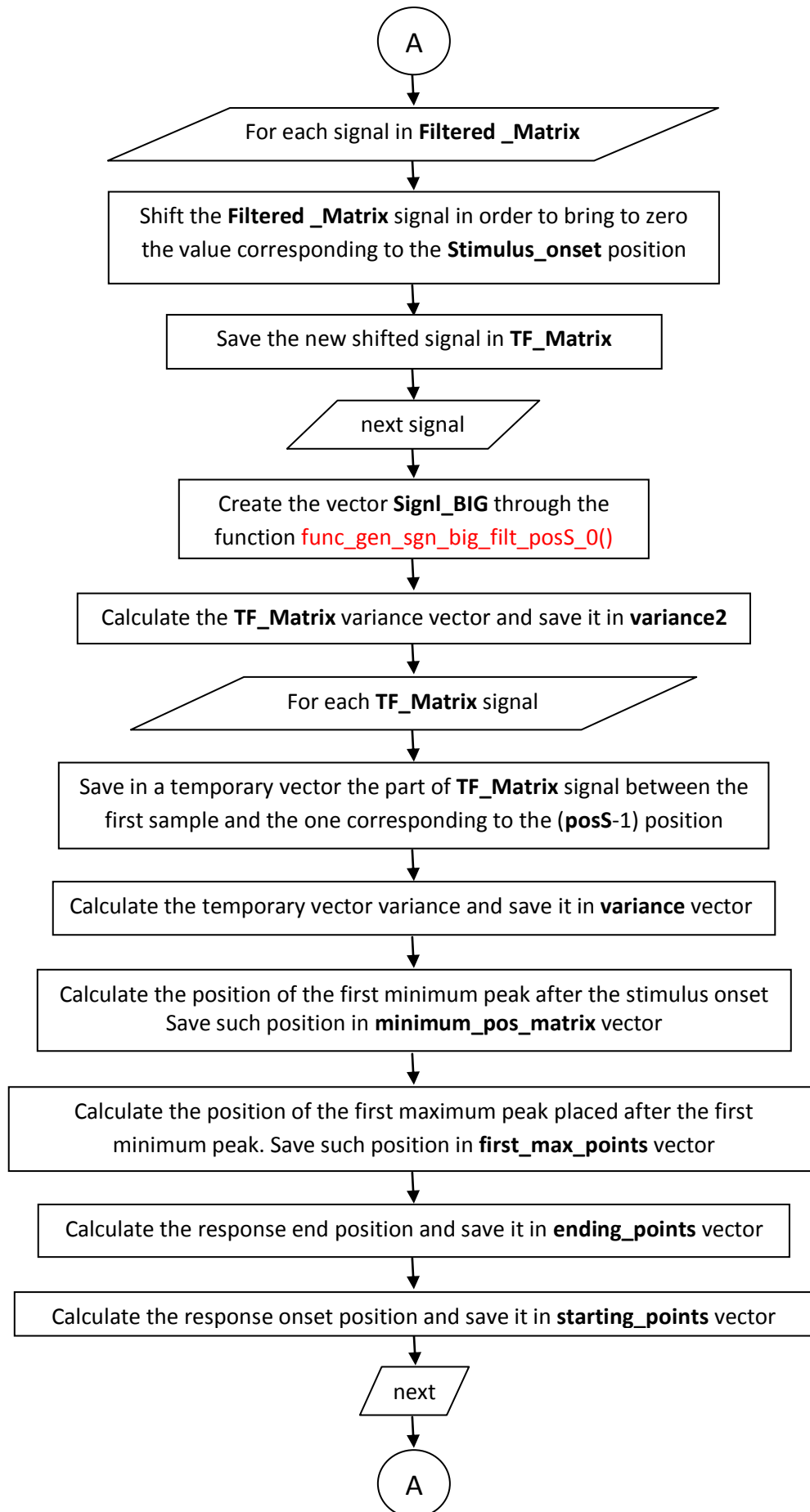
The following flowchart shows the control flow of the `CallerFunction()` function.

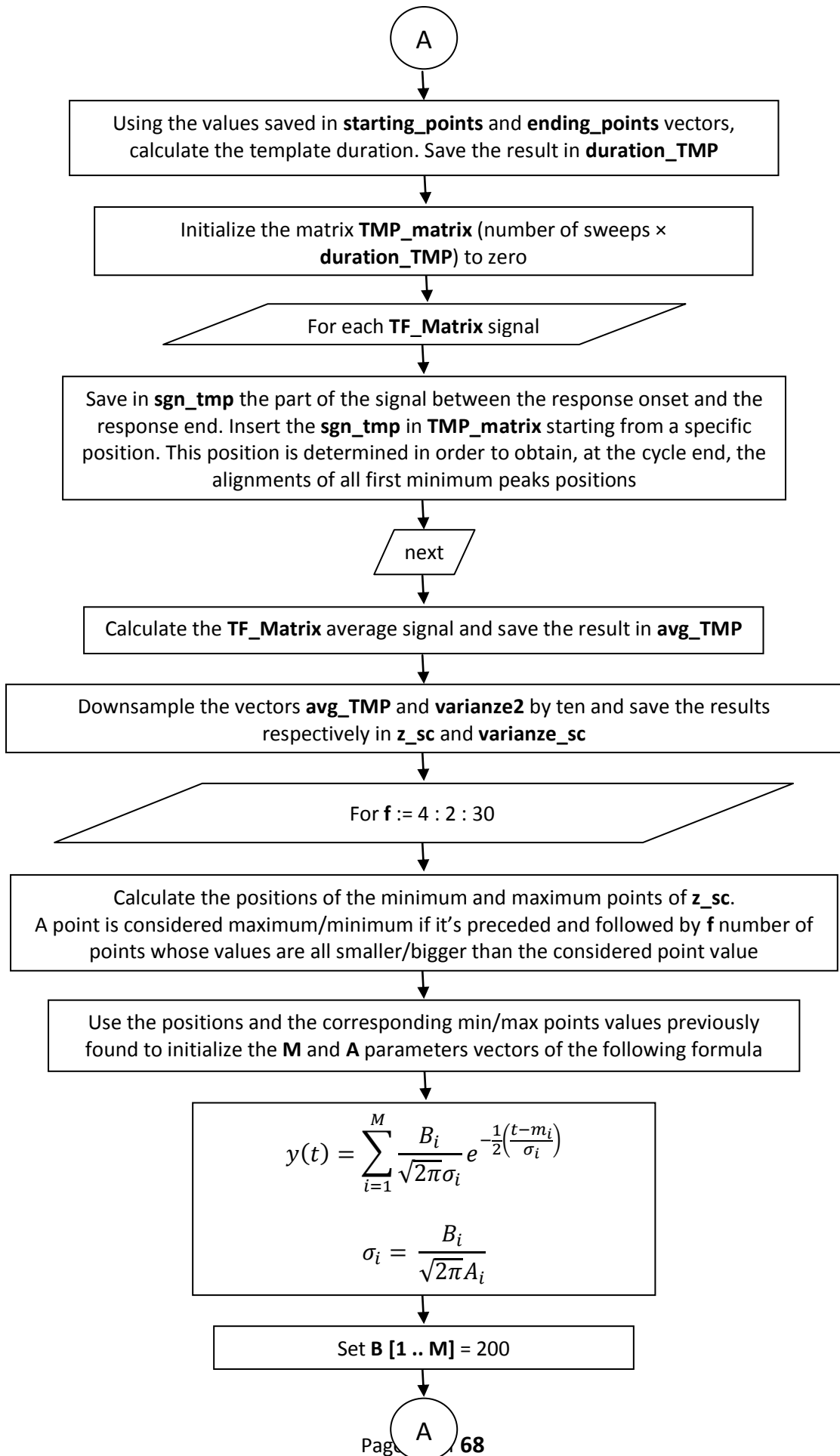


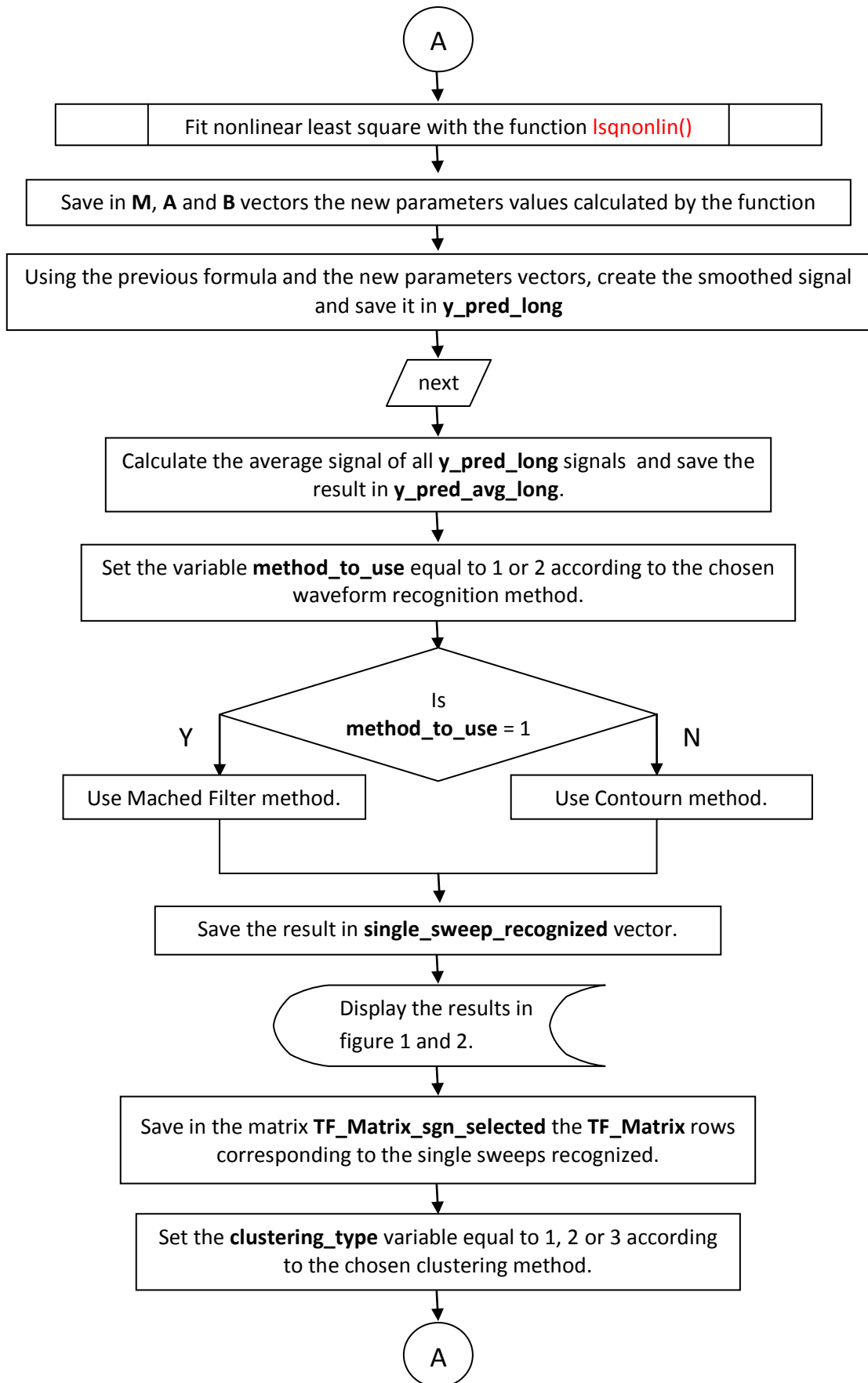
3.2.1.2 FUNC_ANALYSIS_SINGLE_SWEEP

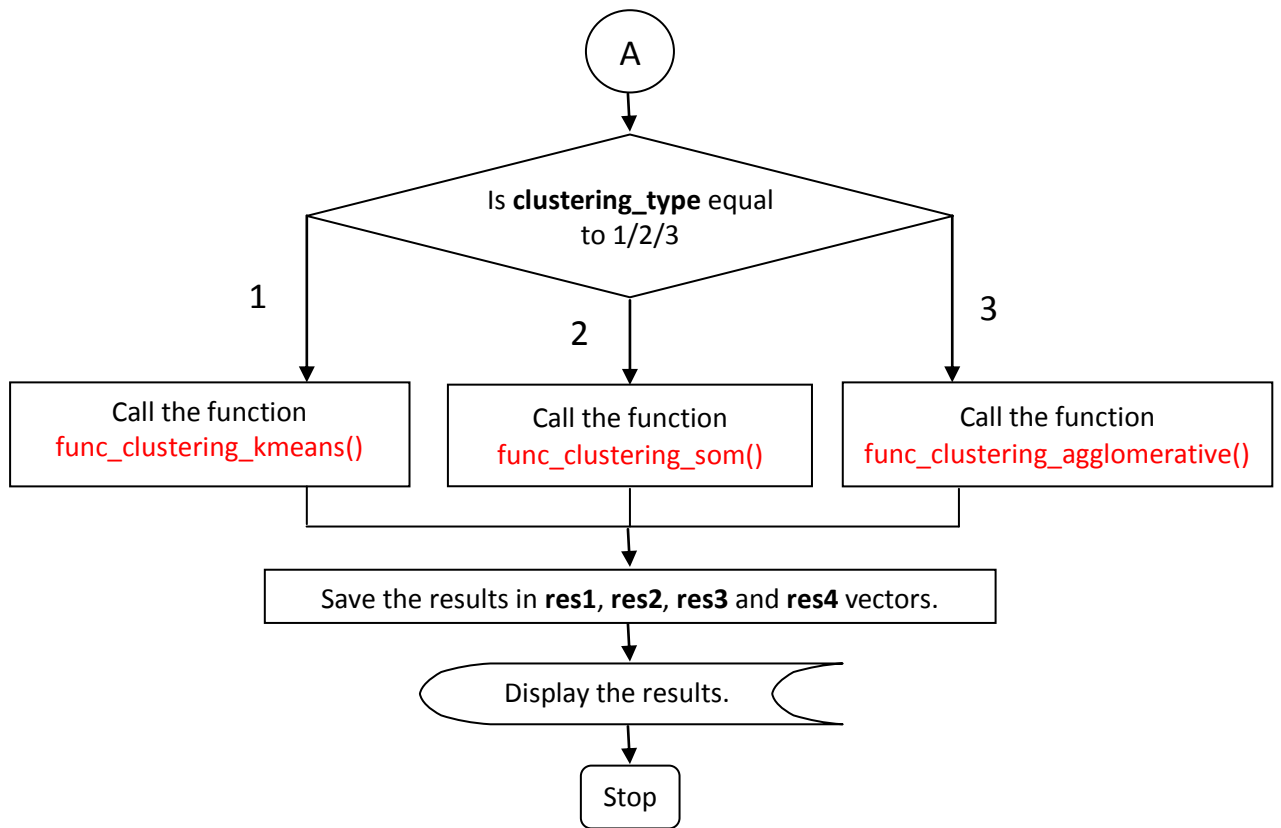
The following flowchart shows the control flow of the Funz_Analysis_Single_Sweep() function.











3.2.2 ALGORITHM OF THE METHOD

3.2.2.1 FUNCTION: `func_gen_sgn_big_filt_posS_0 ()`

Input: TF_Matrix.

Output: Segnale_BIG_filtrato_posS_0.

Method:

- Save TF_Matrix dimensions in the variables nr and nc.
- Initialize Segnale_BIG_filtrato_posS_0 vector, whose length is given by the product of nr and nc.
- Set pos_iniziale = 1.
- For i = 1 : nr
 - Set pos_finale = pos_iniziale + nc - 1.
 - In Segnale_BIG_filtrato_posS_0, starting from pos_iniziale until pos_finale, save the i-th row of TF_Matrix.
 - Set pos_iniziale = pos_finale + 1.
- End.
- Return Segnale_BIG_filtrato_posS_0.

3.2.2.2 HOW TO CALCULATE THE VECTOR **minimum_pos_matrix**

Input: TF_Matrix.

Output: minimum_pos_matrix.

Method:

- Find the sample position closer to the time instant = 0.2 ms through the **closer_point()** function; save the result in pos_02.
- Initialize the minimum_pos_matrix vector.
- For a = 1 : number of sweeps
 - Save in sgn_tmp the samples corresponding to the positions between posS and pos_02.
 - Find, in sgn_tmp, the minimum point position. Add (posS-1) to such value and save the result in a-th position of minimum_pos_matrix vector.
- End.

3.2.2.3 FUNCTION: **closer_point ()**

Input: a vector, value.

Output: closer_pos.

Method:

- Set N equal to the vector length.
- Initialize the diff vector.
- For a = 1 : N
 - Save in the a-th position of diff the absolute value of the difference between the a-th sample of vector and the value variable.
- End.
- Find the position of the minimum sample stored in the diff vector and save it in the closer_point variable.
- Return closer_point.

3.2.2.4 HOW TO CALCULATE THE VECTORS **first_max_points, ending_points**

Input: TF_Matrix, minimum_pos_matrix.

Output: first_max_points, ending_points.

Method:

- Initialize the first_max_points and the ending_points vectors.
- For c = 1 : number of sweeps
 - Initialize the vet_pos_ok matrix.
 - Save in sgn_tmp the samples corresponding to the positions between the c-th minimum_pos_matrix value and the final sample.
 - Set p = 1.
 - For a = 701 : (length(sgn_tmp) - 700)
 - If the a-th sample value of sgn_tmp vector is preceded and followed by 700 points whose values are smaller than such value, save:

- the a position value, in the 1st column of the p-th row of vet_pos_ok matrix,
- the value 2 in the 2nd column,
- the a-th value of sgn_tmp, in the 3rd column.
- p = p+1.
- else, if the a-th sample of sgn_tmp vector is preceded and followed by 700 points whose values are bigger than such sample, save:
 - the a position value, in the 1st column of the p-th row of vet_pos_ok matrix,
 - the value 1 in the 2nd column,
 - the a-th value of sgn_tmp, in the 3rd column.
 - p = p+1.
- End.
- End.
- Call `elimina_punti_specifici()` function and save the results in `vett_pos_ok_max` and in `vett_pos_ok_min` vectors.
- Save in the c-th `first_max_points` position the first value stored in `vett_pos_ok_max`.
- Create the matrix `vett_MinMax_ord`.
Save in the 1st column the positions of the ascending ordered maximum and minimum points.
Save in the 2nd column the value 1 or 2, which defines if such point corresponds, respectively, to a minimum or a maximum.
- Save in the m variable the number of rows of `vett_MinMax_ord`. Set `m = 5`, if the number of rows is bigger than 4.
- Save the m value in the c-th position of `num_MinMax_point` vector.
- Call `func_ricerca_fine_risposta()` function and save the results in `pos_best_next_to_zero`.
- Save in the c-th position of `ending_points` vector:
`pos_best_next_to_zero + minimum_pos_matrix(c) - 1`.
- End
- Add to the `first_max_points` vector the `minimum_pos_matrix` vector to obtain the correct maximum positions.

3.2.2.5 FUNCTION `elimina_punti_specifici()`

Input: `vett_pos_ok`.

Output: `pos_ok_max`, `pos_ok_min`.

Method:

- In order to obtain a sequence of maximum and minimum points, remove, if necessary, the unwanted rows of `vett_pos_ok`. The sequence must start with a maximum point.
 - If there is a sequence of minimum points, remove all the rows except the row corresponding to the smallest minimum point.

- If there is a sequence of maximum points, remove all the rows except the row corresponding to the biggest maximum point.
- Return in output: the maximum points positions selected, stored in `pos_ok_max`, and the minimum points positions selected, stored in `pos_ok_min`.

3.2.2.6 FUNCTION `func_ricerca_fine_risposta()`

Input: `sgn_tmp`, `vett_MinMax_ord`, `m`.

Output: `pos_best_next_to_zero`.

Method:

- Save in `pos_iniziale` the value corresponding to the `m`-th row, 1st column of `vett_MinMax_ord` matrix.
- Extract the part of `sgn_tmp` vector between (`pos_iniziale + 1`) and the last sample. Save the result in `segnale_tagliato`.
- If `segnale_tagliato` first element is a negative value:
 - if `sgn_tagliato` crosses the zero line (doesn't remain negative).
 - Find the position of the first element equal or bigger than zero.
 - Add to `pos_iniziale` the position found and save the result in the variable `pos_best_next_to_zero`.
 - Else
 - Find the position of the element whose value is closer to zero.
 - Add to `pos_iniziale` the position found and save the result in the variable `pos_best_next_to_zero`.
- Else
 - If `sgn_tagliato` crosses the zero line (doesn't remain positive).
 - Find the position of the first element equal or smaller than zero.
 - Add to `pos_iniziale` the position found and save the result in the variable `pos_best_next_to_zero`.
 - Else
 - Find the position of the element whose value is closer to zero.
 - Add to `pos_iniziale` the position found and save the result in the variable `pos_best_next_to_zero`.
 - End
- End
- Return `pos_best_next_to_zero`.

3.2.2.7 HOW CALCULATE THE VECTOR `starting_points`

Input: `TF_Matrix`, `minimum_pos_matrix`.

Output: `starting_points`.

Method:

- Initialize the `starting_points` vector.
- For `a = 1` : number of sweeps

- Save in `sgn_tmp` the samples corresponding to the positions between `posS` and the `a`-th value saved in `minimum_pos_matrix`.
- Find the position of `sgn_tmp` maximum value and save it in `pos_max_tmp`.
- Set `pos_min_tmp` equal to one.
- If `pos_max_tmp` is not equal to one
 - Save in `sgn_tmp2` the `sgn_tmp` samples corresponding to the positions between 1 and `pos_max_tmp`.
 - Find the position of `sgn_tmp2` minimum value and save it in `pos_min_tmp`.
- End.
- Save in the `a`-th position of `starting_points`: `pos_min_tmp + posS - 1`.
- End.

3.2.2.8 CALCULATING THE **single_sweep_recognized** VECTOR WITH **matched filter**

Input: `y_pred_avg_long`, `Segnale_BIG`.

Output: `single_sweep_recognized`.

Method:

- Save the transpose of the vector `y_pred_avg_long` and of the vector `Segnale_BIG` respectively in `TMP_fit` and `sgn_tmp`.
- Flip the columns of `TMP_fit` in the left/right direction (function `fliplr()`).
- Set `N_TMP` equal to the `TMP_fit` length.
- For `a = N_TMP : length(sgn_tmp)`
 - Save in `x_tmp` the `sgn_tmp` samples corresponding to the positions between $(a - N_TMP + 1)$ and `a`.
 - Flip the rows of `x_tmp` in the up/down direction (function `flipud()`).
 - Multiply the vectors `TMP_fit` and `x_tmp`. Save the result in the `a`-th position of `y_matched`.
- End.
- Set `N_BIG` equal to `y_matched` length.
- Set `f = 100` and `k = 1`.
- For `a = (f + 1) : (N_BIG - f)`
 - If the `a`-th sample value of `y_matched` is preceded and followed by `f` points whose values are smaller than such value.
 - Save in the `k`-th column of `vett_pos_massimi` the value of `a`.
 - Set `k = k + 1`.
 - End.
- End.
- Save in the vector `pos_sf` the sequence from `nc` until `nr*nc` with step equal to `nc`, where `nc` and `nr` are, respectively, the single sweep length and the total number of sweeps analyzed.
- Initialize the vector `point_closer_to_end`.
- Set `k = 1`.
- For `a = 1 : length(pos_sf)`
 - Save in `pos_end_tmp` the `a`-th value of `pos_sf`.

- Find the vett_pos_massimi value closer to pos_end_tmp and save it in the k-th position of point_closer_to_end.
- Set k = k+1.
- End.
- Set best_threshold = 0, k = 1, Prob_correct_assignment_old = 0, VP_num_best = 0.
- For b = -500 : 1500
 - Set VP_num = 0, FP_num = 0, FN_num = 0, VN_num = 0, threshold = b-th value.
 - For a = 1 : vett_pos_massimi length
 - Save in pos_max_tmp the a-th value of vett_pos_massimi.
 - Save in val_corrisp the y_matched value corresponding to the position equal to pos_max_tmp.
 - If val_corrisp >= threshold
 - If pos_max_tmp is one of the values stored in point_closer_to_end.
 - Set VP_num = VP_num + 1.
 - Else
 - Set FP_num = FP_num + 1.
 - End.
 - Else
 - If pos_max_tmp is one of the values stored in point_closer_to_end.
 - Set FN_num = FN_num + 1.
 - Else
 - Set VN_num = VN_num + 1.
 - End.
 - End.
 - End.
 - Set Prob_correct_assignment = (VP_num + VN_num) / (vett_pos_massimi length).
 - Set k = k + 1.
 - If Prob_correct_assignment > correct_assignment_old
 - Set VP_num_best = VP_num, best_threshold = threshold, Prob_correct_assignment_old = Prob_correct_assignment.
 - End.
- End.
- Set k = 1.
- Initialize the vector pos_ist_rec.
- For a = 1 : (point_closer_to_end length)
 - Save in val_corrisp the y_matched value stored in the position corresponding to the a-th value of point_closer_to_end.
 - If val_corrisp >= best_threshold
 - Save in the k-th position of pos_ist_rec the a-th value of point_closer_to_end.
 - Set k = k + 1.
 - End.
- End.

- Divide the pos_ist_rec vector for nc, round the results and save it in the vector single_sweep_recognized.
- Return single_sweep_recognized.

3.2.2.9 CALCULATING THE **single_sweep_recognized** VECTOR WITH THE **contour**

Input: y_pred_avg_long, Segnale_BIG.

Output: single_sweep_recognized.

Method:

- Save the transpose of the vector y_pred_avg_long in the vector TMP_fit.
- Set N_TMP equal to the TMP_fit length.
- Save in the vector pos_sf the transpose of the sequence from nc until nr*nc with step equal to nc. The variables nc and nr are, respectively, the single sweep length and the total number of sweeps analyzed.
- Initialize the vector VTMP.
- For i = 1 : (TMP_fit length)
 - Set summation = 0.
 - For j = 1 : nr
 - Save in sgn_tmp the j-th row of TMP_matrix.
 - Set summation = summation + (sgn_tmp(i) – TMP_fit(i))^2.
 - End.
 - Save in the i-th position of VTMP the division result between summation and nr.
- End.
- Initialize the vectors Sup_lim and Inf_lim.
- Calculate the square root of the VTMP average value and save it in the variable SD.
- Set a = SD and b = 2*SD.
- For i = 1 : (TMP_fit length)
 - Save in the i-th position of Sup_lim: $TMP_fit(i) + (a * \sqrt{VTMP(i)} + b)$.
 - Save in the i-th position of Inf_lim: $TMP_fit(i) - (a * \sqrt{VTMP(i)} + b)$.
(sqrt means square root).
- End.
- Initialize the vector istant_time_ok.
- Set k = 1.
- For i = N_TMP : (Segnale_BIG length)
 - Save in the vector x_tmp the Segnale_BIG values stored between the positions (i – N_TMP + 1) and i.
 - If (x_tmp >= Inf_lim) && (x_tmp <= Sup_lim)
 - Save in the k-th position of istant_time_ok the value of i.
 - Set k = k + 1.
- End.
- Initialize the vector single_sweep_recognized.
- Set VP_num = 0, FP_num = 0, FN_num = 0, VN_num = 0 and x = 1.
- For i = 1 : (Segnale_BIG length)
 - Set pos = i.

- If pos is equal to one of the values stored in `istant_time_ok` and is equal to one of the values stored in `pos_sf`.
 - Set `VP_num = VP_num + 1`.
 - Save in the x-th position of `single_sweep_recognized` the value of i.
 - Set `x = x + 1`.
- Else, if pos is equal to one of the values stored in `istant_time_ok`
 - Set `FP_num = FP_num + 1`.
- Else, if pos is equal to one of the values stored in `pos_sf`
 - Set `FN_num = FN_num + 1`.
- Else
 - Set `VN_num = VN_num + 1`.
- End.
- Set `Prob_correct_assignment = (VP_num + VN_num) / (Segnale_BIG length)`.
- Set `single_sweep_recognized = single_sweep_recognized / nc`.
- Return `single_sweep_recognized`.

3.2.2.10 FUNCTION `func_clustering_kmeans()`

Input: `TF_Matrix_sgn_selected`.

Output: `mat_coordinate_clu_best`, `mat_appartenenze_best`, `mat_coordinate_clu_old_best`, `num_clu_sel`.

Method:

- Set `mat_coordinate_sgn = TF_Matrix_sgn_selected`.
- Save in the `nr` variable the rows number of `mat_coordinate_sgn`.
- Ask to the user the clusters number. Save the answer in the variable `K`.
- Set the variable `num_clu_sel = K`.
- Ask to the user the iterations number to use. Save the answer in the `max_iter` variable.
- Ask to the user the optimization criteria to use. Save the answer in the `criteria` variable.
- Ask to the user the metric type to use. Save the answer in the `measure` variable.
- Set `variance_old = 10^9`, and `ratio_old = 10^9`.
- Initialize the `mat_coordinate_clu_best`, `mat_appartenenze_best`, `mat_coordinate_clu_old_best` matrices.
- For `h = 1 : max_iter`
 - Call the `func_initial_position_gen()` function. Saves the results in the `mat_coordinate_clu` matrix.
 - Set `mat_coordinate_clu_old = mat_coordinate_clu`.
 - Initialize the `mat_appartenenze` matrix.
 - Save in `mat_appartenenze`:
 - in the 1st column, the sequence that starts from 1 until `nr` with step = 1;
 - in the 2nd column, the sequence of natural numbers randomly generated from a uniform distribution between 1 and `K`.
 - Set `mat_appartenenze_old = mat_appartenenze`.
 - Set `condizione = 1` and `iterazioni = 1000`.

- While condizione = 1
 - For i = 1 : iterazioni
 - Call the `func_Aggiorna_appartenenze()` function. Save the results in the `mat_appartenenze` and `mat_coordinate_clu` matrices.
 - If `mat_appartenenze_old = mat_appartenenze`.
 - Set `condizione = 0` and breaks the for cycle.
 - Else
 - Set `mat_appartenenze_old = mat_appartenenze`.
 - End.
 - End.
 - Set `condizione = 0`.
- End.
- If `criteria = 1`
 - Call `func_variance_calc()` function. Save the result in `variance` variable.
 - If `variance < variance_old`
 - Set:
 - `mat_coordinate_clu_best = mat_coordinate_clu,`
 - `mat_coordinate_clu_old_best = mat_coordinate_clu_old,`
 - `mat_appartenenze_best = mat_appartenenze,`
 - `variance_old = variance;`
 - End.
- Else, if `criteria = 2`
 - Call `func_variance_calc_new()` function. Save the result in `variance` variable.
 - If `variance < variance_old`
 - Set:
 - `mat_coordinate_clu_best = mat_coordinate_clu,`
 - `mat_coordinate_clu_old_best = mat_coordinate_clu_old,`
 - `mat_appartenenze_best = mat_appartenenze,`
 - `variance_old = variance;`
 - End.
- Else, if `criteria = 3`
 - Call `func_variance_entro()` function. Save the result in `variance_entro` variable.
 - Call `func_variance_tra()` function. Save the result in `variance_tra` variable.
 - Save in the `ratio` variable the division results between `variance_entro` and `variance_tra`.
 - If `ratio < ratio_old`
 - Set:
 - `mat_coordinate_clu_best = mat_coordinate_clu,`
 - `mat_coordinate_clu_old_best = mat_coordinate_clu_old,`
 - `mat_appartenenze_best = mat_appartenenze,`
 - `ratio_old = ratio;`
 - End.

- End.
- End.
- Return the `mat_coordinate_clu_best`, `mat_coordinate_clu_old_best`, `mat_appartenenze_best` matrices and the `num_clu_sel` variable.

3.2.2.11 FUNCTION `func_clustering_som()`

Input: `TF_Matrix_sgn_selected`.

Output: `mat_coordinate_clu_new`, `mat_coordinate_clu_old`, `mat_assegnamenti_finali`, `num_clu_sel`.

Method:

- Set `mat_coordinate_sgn = TF_Matrix_sgn_selected`.
- Save in the `nr` variable the rows number of `mat_coordinate_sgn`.
- Ask to the user the clusters number. Save the answer in the variable `K`.
- Set the variable `num_clu_sel = K`.
- Ask to the user the iterations number to use. Save the answer in the `max_iter` variable.
- Ask to the user the eta parameter value to use. Save the answer in the `eta_max` variable.
- Ask to the user the metric type to use. Save the answer in the `measure` variable.
- Save in `vet_indici` vector the sequence that starts from 1 until `nr` with step = 1.
- Call the `func_initial_position_gen()` function and save the result in `mat_coordinate_clu` matrix.
- Set `mat_coordinate_clu_old = mat_coordinate_clu`.
- For `i = 1 : max_iter`
 - Set `eta = eta_max * (1 - (i / max_iter))`.
 - Call the `func_sgn_selection()` function and save the result in `sgn_selected_tmp` variable.
 - Save in `sgn_tmp` the `mat_coordinate_sgn` row corresponding to `sgn_selected_tmp` value.
 - Call the `func_closer_clu()` function and save the result in the `winner` variable.
 - Call the `func_aggiornamento_new()` function and save the result in the `mat_coordinate_clu` matrix.
- End.
- Call the `func_assegnamento_sgn()` function and save the result in the `mat_assegnamenti_finali` matrix.
- Set `mat_coordinate_clu_new = mat_coordinate_clu`.
- Return the `mat_coordinate_clu_new`, `mat_coordinate_clu_old`, `mat_assegnamenti_finali` matrices and the `num_clu_sel` variable.

3.2.2.12 FUNCTION `func_clustering_agglomerative()`

Input: `TF_Matrix_sgn_selected`.

Output: `cluster_mat`, `cluster_vet_stop`, `vect_distance`, `K`, `mat_dendrogram`.

Method:

- Ask to the user the metric type to use. Save the answer in the `measure` variable.
- Call the `func_calcolo_mat_dist_new()` function and save the result in the `mat_distance` matrix.
- Save in `nr` and `nc`, respectively, the rows and columns numbers of `mat_distance`.
- Initialize the `mat_dendrogram` matrix.

- Ask to the user the `mat_distance` upgrade rule to use. Save the answer in the rule variable.
- Ask to the user the clusters number. Save the answer in the variable `K`.
- If `K` is not equal to zero
 - Set `stop_iter = nr - K + 1`.
- Else
 - Set `stop_iter = nr`.
- End.
- Set `k = 1`.
- Initialize the `cluster_mat` cell matrix and `cluster_vet` cell array.
- For `i = 1 : nr`
 - Save in the `i`-th position of `cluster_vet` the string containing the `i` value.
 - For `j = 1 : nc`
 - If `j = 1`
 - Save in the `cluster_mat` `i`-th row, `j`-th column the string containing the `k` value.
 - Set `k = k + 1`.
 - Else
 - Save in the `cluster_mat` `i`-th row, `j`-th column the string containing an empty space.
 - End.
 - End.
- End.
- Initialize the `vect_distance` vector and the `cluster_vet_stop` cell array.
- Save in `vect_dendro_sup` vector the sequence starting from 1 until `nr` with step = 1.
- For `i = 2 : nr`
 - Call the `func_aggiornamento_mat_dist()` function and save the results in: `mat_distance_new` matrix, `cluster_vet_new` cell array, `dist_sel` variable, `vect_dendrogram` and `vect_dendro_sup` vectors.
 - Save `dist_sel` variable in the `i`-th position of `vect_distance` vector.
 - Set `mat_distance = mat_distance_new`.
 - For `j = 1 : (cluster_vet_new length)`
 - Save in the `cluster_mat` `i`-th row, `j`-th column the `j`-th string stored in `cluster_vet_new`.
 - End.
 - If `i = stop_iter`
 - Set `cluster_vet_stop = cluster_vet_new`.
 - End.
 - Set `cluster_vet = cluster_vet_new`.
 - Save `vect_dendrogram` vector in the `mat_dendrogram` (`i-1`)-th row.
 - Set `vect_dendro_sup = vect_dendro_sup_new`.
- End.

- If $K = 0$
 - Initialize the cluster_vet_stop cell array and the vect_diff vector.
 - Set $k = 1$.
 - For $i = 2 : (\text{vect_distance length})$
 - Save in vect_diff k-th position the difference result between the vect_distance(i) value and the vect_distance(i-1) value.
 - Set $k = k + 1$.
 - End.
 - Set brantch_longer variable equal to the maximum value position of vect_diff.
 - Set stop_iter = brantch_longer + 1.
 - Set $k = 1$.
 - For $i = 1 : (\text{nr} - \text{stop_iter} + 1)$
 - Save in cluster_vet_stop i-th position the string stored in the i-th row, p-th column of cluster_mat. Where p corresponds to the stop_iter value.
 - Set $k = k + 1$.
 - End.
- End.
- Set $K = \text{cluster_vet_stop length}$.
- Return the cluster_mat cell matrix, the cluster_vet_stop cell array, the vect_distance vector, the K value, the mat_dendrogram matrix.

3.2.3 DESCRIPTION OF THE METHOD

The `func_Analysis_Single_Sweep()` function is composed by four main steps which are described in the subsequent subsections.

- Pre-processing.
- Smoothing.
- Waveform recognition.
- Clustering.

3.2.3.1 PRE-PROCESSING

- The single sweep is initially filtered with a low-pass digital Butterworth filter to eliminate the high frequency noise. The filter coefficients are calculated by the `butter()` function. These coefficients are used as parameters of an ARMA model through which the `filtfilt()` function calculate the filtered signal. The optimal filter order is found as 5 and the cutoff frequency as 1000 Hz.

- The filtered single sweep is translated in order to start from a zero value the evaluation of the signal after the stimulus onset. Another reasonable way is to set to zero the average of the signal part before the stimulus onset, but it was seen that this way doesn't lead at the same good final results.
- Using the `funz_gen_sgn_big_filt_posS_0 ()` function is generated a single signal, saved in the `Segnale_BIG` vector, which contains the sequence of the filtered and translated sweeps.
- At this point four main points are searched in the filtered and translated single sweep.
- The points are: the response onset, the first minimum valley, the first maximum peak and the response end points.
- It was seen that the signal, after the stimulus onset instant, behaves in two main ways. In the first case, the signal presents a little valley followed by a little peak before starting the downhill toward the first minimum valley point. In this case the response onset instant is set equal to the little valley point instant.
- In the second case, the signal starts suddenly, more or less, the downhill and doesn't present any little peak. In this case the program fixes the response onset equal to the stimulus onset instant even if a better choice is given by the recognition of the point of maximum gradient variation. Due to the aim of this point research, however, the program choice is acceptable.
- It was seen that the first minimum valley point is present inside the time interval [0.15, 0.2 ms]. This fact has been proved by a single sweeps visual examination and by a-priori knowledge.
- So, through a simple research inside this time interval, it is easy to find the minimum valley point.
- For the detection of the last two principal points many different methods have been used, based on the gradient or on the signal first derivate, but it was seen that the best one is the following.

- At first the method identifies the min/max point present in the signal part between the minimum valley point sample and the last sample.
- A point is considered maximum/minimum if it's preceded and followed by a number x of points whose values are all smaller/bigger than the considered point value. The value of x is fixed by the program equal to 700 that corresponds to a time interval of 0.035 ms.
- The first maximum peak is defined as the first maximum point of the series.
- Depending on the min/max series length is selected a min/max point. After the point position selected starts the research of the response end point position that corresponds to the first point position closer to the zero value.
- To recognize the presence of a particular waveform inside a noisy signal in both the methods used it's necessary to create a template signal. To generate the template are used the response onset and the response end point position. The signal part between these two points is extracted, and saved in the template matrix. All the first minimum valley points of the cut signals are aligned. Then, a zero-padding operation is used to obtain aligned signals of the same length. At the end an average operation on the template matrix generates the template.

3.2.3.2 SMOOTHING.

- It was seen that if the calculated template is used directly in the waveform recognition methods, the results obtained are unsatisfactory probably due to small oscillations presents inside the template. So it was decided at first to downsample the template and then to smooth it.
- To smooth the template a sum of gaussian equation is used to initially approximate the signal.
- The initial values of the parameters equations are sets equal to the min/max points found inside the template with the method presented above. Using the `lsqnonlin()` function are calculated the best parameters values of the smooth signal.
- The `lsqnonlin()` function requires to define the noise standard deviation (SD). We tried to use a SD or CV (coefficient variance) constant value but the best solution is

given by the SD vector of the aligned signal matrix. Using it, the weighted residuals obtained are quite good due to they are uncorrelated and within the range of -1 and +1.

- The min/max points recognition method requires to fix the x value. The x value determines the min/max points number and so the parameters number. Through a cycle it was found, using the Akaike criterion, the best x value, but it was decided to average all smoothed template, calculated for all x different values, to reduce the template variability.
- Using the new parameters values the smooth signal, as long as the single sweep signal, is calculated. This new template is now ready to be used in the waveform recognition methods.

3.2.3.3 WAVEFORM RECOGNITION METHODS

- The two methods implemented by the program are the matched filter method and the contour method. At the beginning the program asks to the user to select the method to use.
- Both the methods use the smooth template and the signal stored in the Segnale_BIG vector.
- The matched filter method generates in output a signal that should presents in correspondence of the maximum peak instants the waveform occurrence instance.
- In practice are found more peaks than the sweeps number. Moreover there is a little latency between the peaks instances and the waveform occurrence instances. So, after the peaks recognition, for each maximum peak instant, it is recognized the waveform occurrence instant closer to it.
- Then is calculated the best threshold that maximize the correct assignment probability value witch depends on the TP and FP values. A peak point is considered a TP (true positive) or a FP (false positive) if its value is bigger than the threshold, in the other case it can be a TN (true negative) or a FN (false negative). The peak is considered a TP if it is the only one closer to a waveform occurrence instant, if not it's considered a FP. A peak, at the same way, is considered a TN or a FN.

- At the end, using the best threshold found, are saved the positions of all waveform occurrence instant corresponding to the TP maximum points. The TP points number corresponds to the number of sweeps recognized.
- The contour method at first calculates, using the smooth template (TMP) and the template matrix, VTMP vector in which are saved the TMP samples variances.
- Then the upper and lower bounds vectors are calculated, they depend on the TMP and VTMP vectors and also on the a and b values. These two values determinate the shape and the width of the contour.
- For each signal sample of Segnale_BIG, starting from the D-th position, where D is set equal to the TMP length, if it is preceded by D points, whose values are located between the two bounds, so its position is considered a waveform occurrence instant and saved in the instant_time_ok vector. It was seen that the points number selected in this way is bigger than the single sweep number so an ulterior selection step is necessary.
- Knowing the real waveform occurrence instants it is possible to consider a Segnale_BIG sample as a TP if its position is present in the instant_time_ok vector and if it is a waveform occurrence instants. If it is true only the first condition the sample is considered a FP. If it is true only the second condition, the sample is considered a FN, while if both the conditions are false the sample is considered a TN.
- Using the TP sample it is easy to select the single sweeps recognized.
- The a and b values influence the contour shape and so the final results. A possible solution to set these parameters is to find those values witch maximize the correct assignment probability value. It was seen this way is very time consuming, so it was decided to set the a value equal to the root square of the VTMP average value, while the b value is set four time bigger than the a value.
- At the end both the methods return in output the single_sweep_recongized vector in which are stored numbers, between 1 and 100, corresponding to the single sweep selected.

3.2.3.4 CLUSTERING

- Three different clustering techniques are implemented in this method. These techniques are the k-means, the SOM (self organizing maps) and the hierarchical agglomerative.
- All these techniques require in input the Parameters_matrix matrix in which are stored the coordinates that define the selected single sweeps positions in the M-dimensional space.
- Two ways have been tried to define the coordinates to use.
- The first one uses as coordinates the four main point values (response onset, first minimum valley, first maximum peak and end of response points) and their corresponding time instants ($M = 8$).
- The second way selects a fixed number of points ($M = x$) starting from the smallest first maximum peak instants and uses them as coordinates. It was tried to set x equal to 200 or 400.
- Initially, all these techniques ask to the user to fix the parameters values (metric type, output number of clusters etc.). In this way it is possible to find the best setup to obtain the best results.
- The k-means method starts calculating the initial coordinates of the centroids. Then each signal is associated to the nearest centroid. All associations are saved in a matrix. Then, for a fixed number of iterations: the centroids coordinates are recalculated based on the old associations matrix and a new associations matrix is calculated.
- In case of equality between the old association matrix and the new one the program automatically stops before reaching the last iteration.
- The final result is influenced by the initial centroids coordinates. So, to find the best final solution, are tried many different initial centroids coordinates, and it is selected the one which minimize a selectable cost function.
- This technique returns in output the initial and final centroids coordinates, the associations matrix and the clusters number.

- The SOM method, as the previous one, starts defining the initial coordinates of the centroids (neurons). Then for a fixed number of iterations: a point is randomly selected, it is found his the closest neuron (called winner), the coordinates of the winner and of its neighbors are updated. It was decided to use a linear layout and one step neighborhood.
- At the cycle end each point is associated to the closest neuron.
- This technique returns the initial and final neurons coordinates, the associations matrix and the clusters number.
- The hierarchical agglomerative clustering method starts generating the distances matrix in which are saved the distances between each couple of points. The way to calculate the distance is defined by the metric parameter.
- It is initialized the cluster_mat cells matrix formed by strings of empty space, only the first column contains strings in which are saved the numbers that identify all points (from one until nr, where nr = number of points).
- Then for i starting from 2 until (nr -1): it is find the couple of points closer to each other, they are merged, it is saved the distance between them, it is generated a new distances matrix using one of the three selectable rules and it is updated the cluster_mat cell matrix.
- The hierarchical agglomerative method returns in output the final cluster_mat cell matrix, the cluster_mat column corresponding to the number of clusters selected, the series of distances calculated until the final clusters number is reached, the clusters number and a matrix in which are save the values necessary to create a dendrogram.
- In this technique it is given at the user the possibility to not choose the clusters number which is fixed by default as the clusters number corresponding to the biggest distance, between two points, ever found during the cycle.
- At the end the program uses the output of the clustering techniques chosen to divide the selected single sweeps in groups. The signals of each group are plotted superimposed in the same figure. If it is chosen the hierarchical agglomerative technique it is plotted the dendrogram too.

CHAPTER 4: RESULTS AND DISCUSSION

4.1 RESULTS AND DISCUSSION

In this chapter are reported the results obtained applying the method previously explained.

In figure 12 are reported superimposed all the single sweeps recorded from S1 at the depth of 850 μm and the average single sweep. It is possible to notice the high variability of the single sweeps.

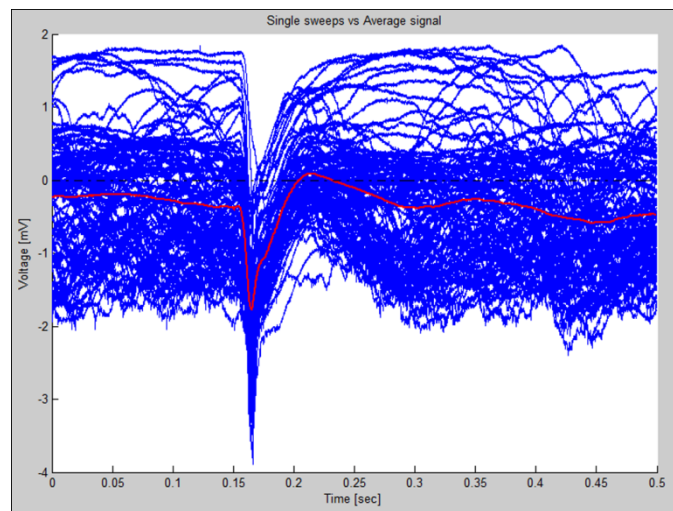


Figure 12. Single sweep versus Average signal.

In figure 13 is reported as example the single sweep number 100. As we can see there is a high frequency spike activity superimposed to the LFP. It appears necessary to apply a low-pass filter to remove it.

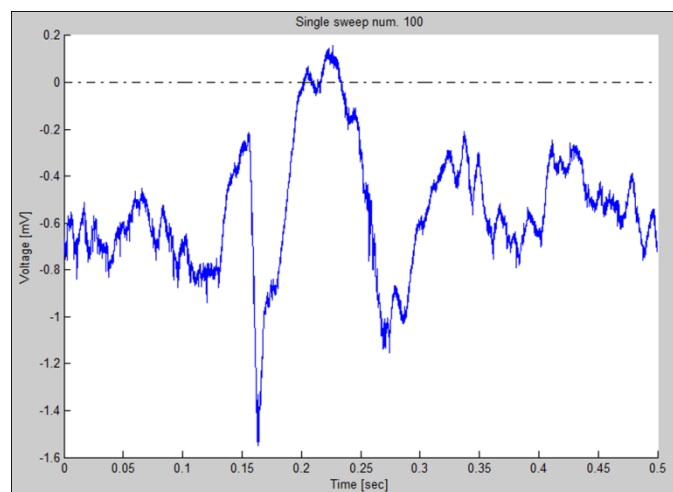


Figure 13. Single sweep number 100.

During the method pre-processing step each single sweep is filtered and translated as to bring to zero the value corresponding to the stimulus onset instant. In figure 14 is reported as example the filtered and translated average signal.

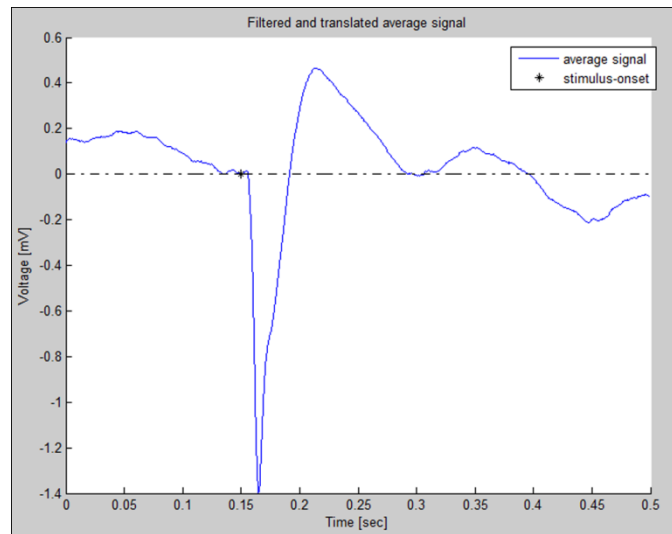


Figure 14. Filtered and translated Average signal.

Another operation made during the pre-processing step is the recognition of the single sweep shape principal points (response onset, first minimum valley, first maximum peak, end of the response points).

In figure 15 is reported as example the filtered and translated single sweep number 100 in which are recognized the four main principal points.

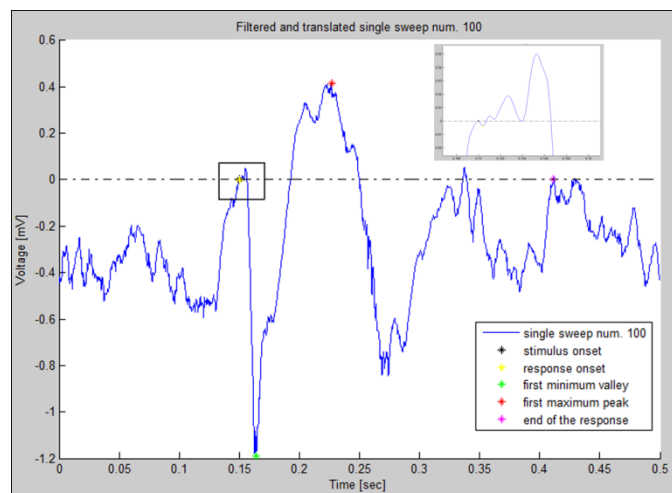


Figure 15. Filtered and translated Single sweep number 100 and principal point recognized. In the top right corner is reported a zoom of the response first part for a better representation of the response onset point recognized.

During the processing step is extract, in each single sweep, the signal part between the response onset and the end of the response instants. Using the cut single sweeps the template is calculated, see figure 16.

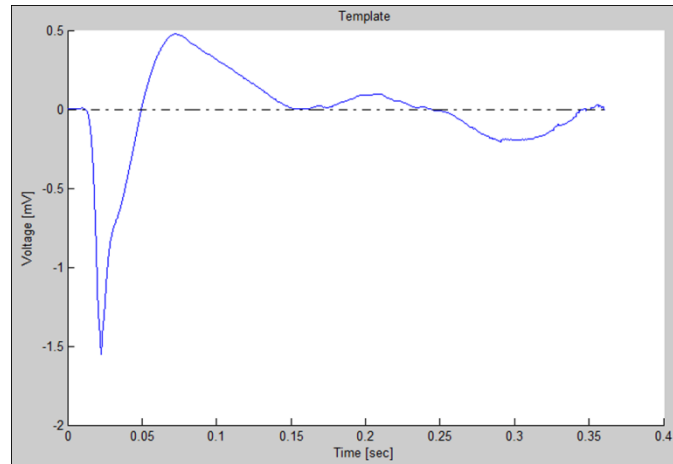


Figure 16. Template.

To reduce the template small oscillations the signal is smoothed. The technique used requires to specify the noise standard deviation (SD). It has been tried to use a constant SD, a variable SD and a constant variance coefficient (CV). For the first case it has been calculated the SD of the part of the filtered and translated average signal between the first sample and the stimulus onset sample. For the second case it has been calculated the SD of the matrix in which are stored the filtered and translated single sweeps while for the third case the CV value has been fixed equal to 0.1. It is possible to notice, by the WRSS values calculated using a fix number of parameters, that the SD variable choice is the best one due to it generates the smallest WRSS value.

	WRSS
Constant SD	13.0791
Variable SD	0.1102
Constant CV	$2.5713 \cdot 10^4$

Table II. Smoothing weighted residuals

Using the Akaike criterion it is possible to determine the best parameters number to smooth the signal but to reduce the smoothed template variability it has been decided to average the smoothed templates obtained using different parameters numbers.

In figure 17 is reported the smoothed template obtained using variable SD and parameters number equals to 75. As we can see the function used to smooth works perfectly.

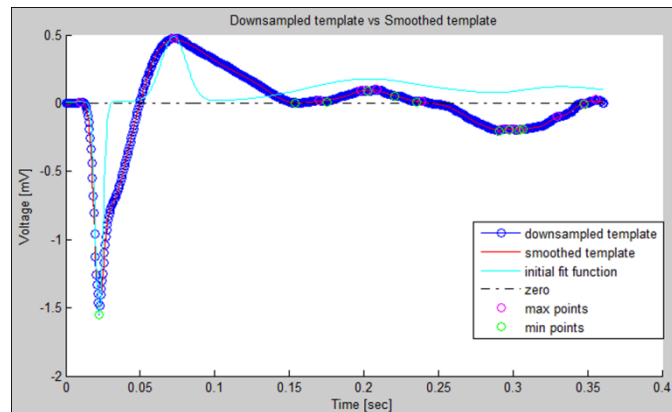


Figure 17. Downsampling template versus Smoothed template.

In figure 18 is reported the average smoothed template superimposed to the original template. As we can see the average smoothed template doesn't fit perfectly the original template but this was planned.

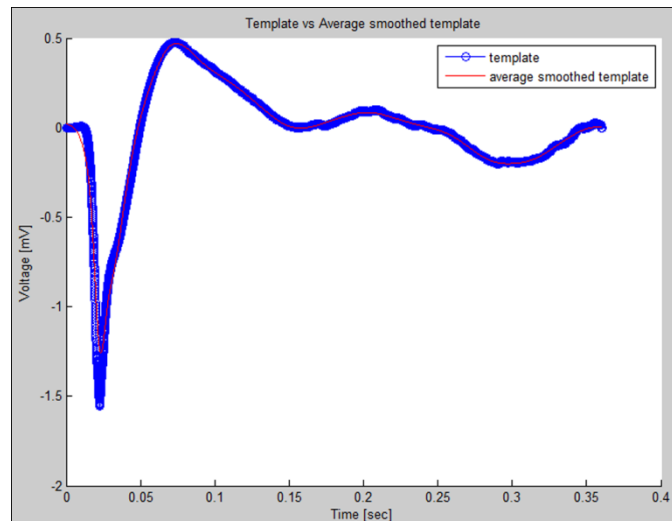


Figure 18. Template versus Average smoothed template

The method proceeds with the waveform recognition step. The average smoothed template is used both by the matched filter and the contour method. In the first method it is calculated the best threshold that maximize the correct assignment probability value (Pc) based on the number of true positive and false positive points found. In the contour method it has been decided to fix the two parameters values (a and b) that determine the upper and lower bounds due to the high computational time required to find the best a and b values that maximize the Pc value. It has been decided to fix the a value equal to the template SD,

while the b equal to three times the a value. Both the methods generates a vector in which are stored the identifiers numbers of the single sweeps recognized. This vector is used in the final method step, clustering. In the figures 19, 20, 21 and 22 are reported the matched filter and the contour method outputs.

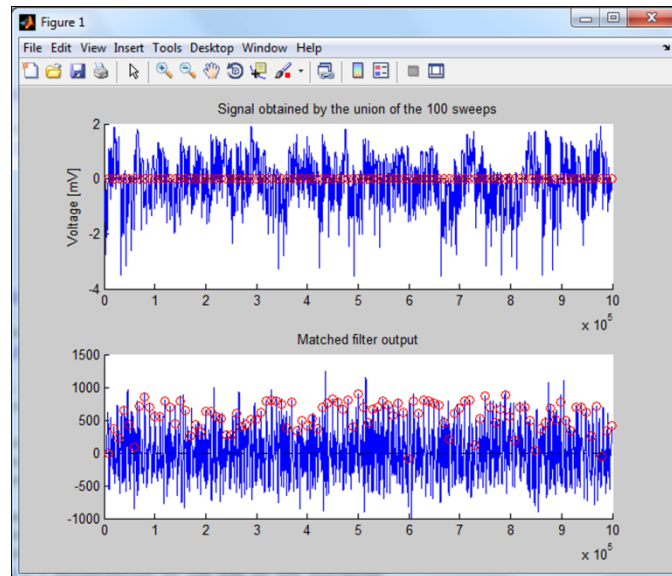


Figure 19. Matched filter output 1. In the top figure is represented the signal obtained by the union of the measured single sweeps, the red circles corresponds to the end of each single sweep. The second figure represents the matched filter output signal; the red circle also here corresponds to the end of each single sweep.

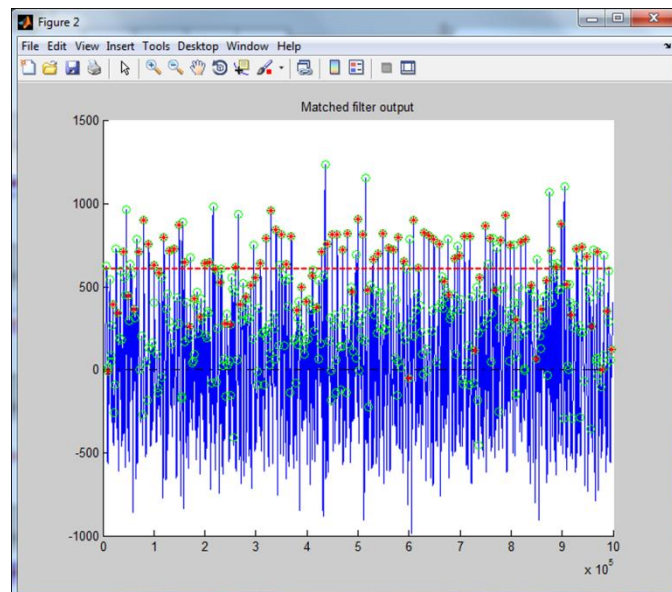


Figure 20. Matched filter output 2. In the figure is reported the matched filter output. The green circles represent the signal peaks, the green circle with a red star are the peaks closer to the end of a single sweep, while the red line corresponds to the best threshold calculated.

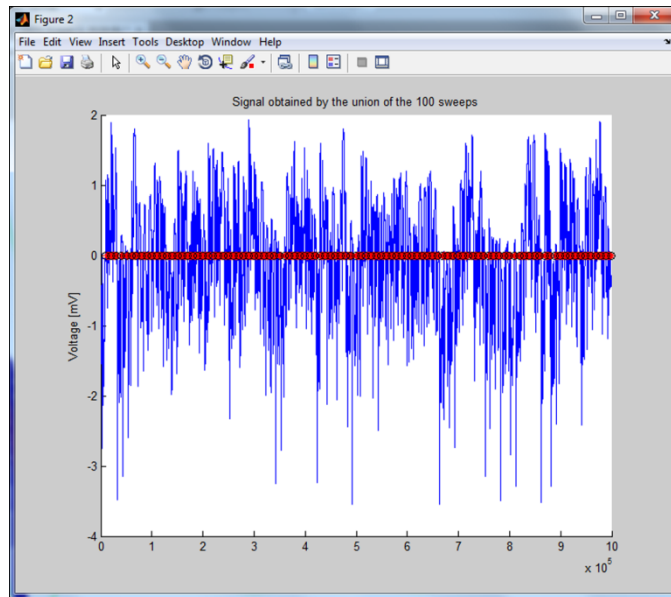


Figure 21. Contour method output 1. In the figure is reported signal obtained by the union of the measured single sweeps, the black circles represents the end of each single sweep while the red stars corresponds to the waveform occurrence instants.

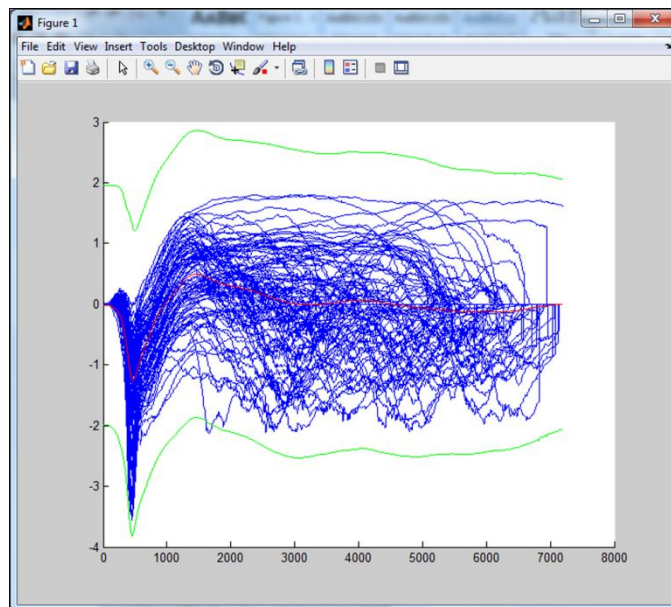


Figure 22. Contour method output 2. In the figure are reported: in blue superimposed the cut single sweeps, in red the average smoothed template and in green the upper and lower bounds.

In this example, due to the higher number of single sweeps recognized by the contour method, it has been decided to use contour method result for the clustering. The Parameters_Matrix, necessary for the clustering, has been calculated using the second way previously presented in section 3.2.3.4 in which the x value has been set to 200. There are

different methods to set the number of clusters as explained in section 2.3.6 but it has been decided in this example to fix the number of clusters equal to ten.

The method lets the user to choose one of the three implemented clustering techniques: K-means, SOM, hierarchical agglomerative. Due to the fact that reporting results from each of the clustering techniques would require much space, we decided to report only the results generated by the k-means technique. Each clustering techniques required to the user to take some choices depending on the clustering technique. In this example it has been decided to set: output clusters number = 10, iterations number = 1000, cost function = minimization of the different clusters variances sum, metric = centered correlation.

In figure 23 the obtained results are reported.

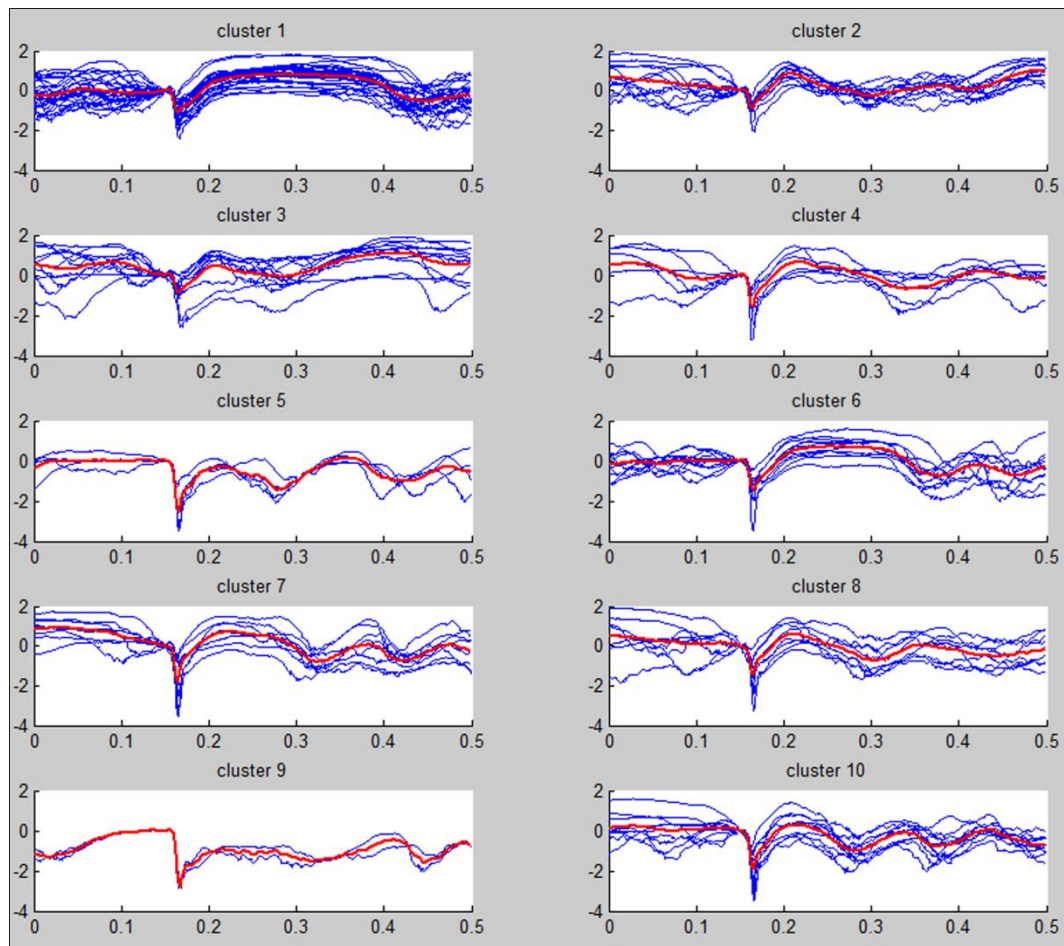


Figure 23. K-means clustering results. In blue are reported the selected single sweeps and grouped in ten different clusters. In red represents the average signal of each cluster.

To prove the method efficiency, using the same clustering techniques and setting, the method has been applied to signals recorded from other seventeen different depths. The results obtained are shown in the following table.

	Cluster no.	Depths [μm]																	
		50	150	250	350	450	550	650	750	850	950	1050	1150	1250	1350	1450	1550	1650	1750
Matched Filter	1	9	9	0	14	0	0	0	0	9	0	0	0	2	0	0	0	0	0
	2	7	2	5	4	3	4	1	4	15	1	1	14	1	6	4	1	5	1
	3	7	3	6	12	2	3	4	5	3	1	1	14	1	3	7	0	2	0
	4	11	3	8	6	10	2	1	4	3	1	1	4	1	1	2	0	3	0
	5	6	6	3	4	2	3	2	4	3	1	1	7	1	3	3	0	5	0
	6	7	6	4	8	3	2	6	5	7	1	1	3	1	2	1	0	2	0
	7	7	6	3	10	5	1	3	2	3	2	1	2	3	1	1	0	3	0
	8	7	2	7	5	9	7	1	2	4	3	1	7	1	5	1	0	2	0
	9	3	5	5	18	3	7	2	2	4	5	2	7	1	0	1	0	0	0
	10	5	5	4	4	3	9	3	0	9	0	0	6	0	0	3	0	0	0
	Total	69	47	45	85	40	38	23	28	60	15	9	64	12	21	23	1	22	1
Contour Method	1	1	2	0	2	25	2	8	6	27	12	12	6	3	22	2	8	13	12
	2	0	6	11	5	2	5	15	11	12	7	14	6	2	21	12	5	11	20
	3	0	2	4	5	7	5	7	14	11	12	5	10	14	16	10	15	11	3
	4	0	2	9	7	4	8	15	10	6	8	5	11	4	15	6	6	6	5
	5	0	1	9	13	6	4	10	7	4	17	10	3	12	20	5	26	3	13
	6	0	4	5	8	18	15	8	8	10	5	3	6	13	25	4	11	9	16
	7	0	1	2	3	7	14	8	10	8	7	16	15	11	3	13	7	13	2
	8	0	2	4	3	2	9	9	2	8	9	15	15	17	2	12	6	8	12
	9	0	2	6	11	1	15	3	18	2	14	12	18	14	11	12	8	10	5
	10	0	0	2	2	2	19	17	14	10	7	8	10	9	4	24	7	16	12
	Total	1	22	52	49	74	96	100	100	98	98	100	100	99	99	100	99	100	100

Table III. Clustering methods results.

4.2 CONCLUSIONS

As seen the low-pass filter used is able to remove all the spike activity, so the filter order and the cutoff frequency have been chosen correctly. Even if it has not been reported, the choice to translate the stimulus-onset of the signal to zero has been found to be better in comparison to the choice to translate the signal to set the average signal's part before the stimulus onset to zero. Infact, in this way more single sweeps are recognized by both the two waveform recognition techniques. Also the choice to smooth the template has been demonstrated better results than to directly use the original template in the waveform recognition.

As Table II shows, the contour method is the best one, this result corresponds to the value of the a and b parameters. Especially the b value influences the results more directly in the sense that, if the value chosen is too high all the sweeps will be recognized. The choice of a and b values has been taken empirically to recognize reasonably high number of sweeps. However, the clustering method works very well also with 100 sweeps.

The generatation of the Prameters_Matrix is done in a way that clusters the single sweeps in a way that reflects their shape. Not all the clustering combinations have been tested so it is not possible to say which one is the best but as we can see from the final results the choices taken are good.

Overall we can conclude saying that the developed method performs well for the type of acquired signals. It has an high efficiency and an acceptable computational time. Since this is the first method which tries to solve the single sweeps classification problem, there is no possibility to compare our results with those obtained through other researches. In the future we hope to extend this method for the signals acquired using high resolution brain-chip interface as described in [26], hoping that our effort would be a guidance for a future leap in understanding the neural network activity in the barrel cortex.

BIBLIOGRAPHY

1. Michael Brecht, "Barrel cortex and whiskey-mediated behaviors", *ScienceDirect* 2007.
2. Ewa Kublik, "Contextual impact on sensory processing at the barrel cortex of awake rat", *Acta Neurobiol Exp* 2004.
3. Woosley T.A., Van Der Loos H., "The structural organization of layer IV in the somatosensory region (SI) of mouse cerebral cortex: the description of a cortical field composed of discrete cytoarchitectonic units", *Brain Research*. 1969, 205-238.
4. Swadlow H.A., "Efferent neurons and suspected interneurons in S1 vibrissa cortex of the awake rabbit: receptive fields and axonal properties", *J. Neurophysiol* 1989, 62: 288-308.
5. Swadlow H.A., "Efferent neurons and suspected interneurons in second somatosensory cortex of the awake rabbit: receptive fields and axonal properties", *J. Neurophysiol* 1991, 66: 1392-1409.
6. Hardingham N., Glazewski S., Pakhotin P., Mizuno K., Chapman P.F., Giese K.P., Fox K., "Neocortical long-term potentiation and experience-dependent synaptic plasticity require alpha-calcium/calmodulin-dependent protein kinase II auto-phosphorylation", *J. Neurosci*. 2003 Jun 1, 23 (11): 4428-36.
7. Diamond M.E., von Heimendahl M., Knutsen P.M., Kleinfeld D., Ahissar E., "Where' and 'what' in the whisker sensorimotor system", *Nat Rev Neurosci*, 2008, 9:601-612.
8. Fox K., "Anatomical pathways. In: Barrel cortex", 2008, pp14-48. Cambridge, UK: Cambridge University Press.
9. C. Cobelli e R. Bonadonna, "Bioingegneria dei sistemi metabolici", *Patron Editore*, 1998 (CB).
10. Rangayyan R.M., "Biomedical Signal Analysis: a case study approach", *IEEE Press*, 2002.
11. <http://www.dei.unipd.it/corsi/bioingegneria/toffolo>
12. North D. O., "An analysis of the factors which determine signal/noise discrimination in pulse carrier systems", *RCA Labs, Princeton, NJ, Rep. PTR-6C*, 1943.
13. Deza E., Deza M. M., "Encyclopedia of Distances", page 94, *Springer*, 2009.
14. Eugene F. Krause, "Taxicab Geometry", *Dover*, 1987.
15. http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/

16. J. L. Rodgers and W. A. Nicewander, "Thirteen ways to look at the correlation coefficient", *The American Statistician*, 42(1): 59-66, February 1988.
17. J. B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations, *Proceedings of the 5-th Berkeley Symposium on Mathematical Statistics and Probability*", Berkeley, University of California Press, 1: 281-297, 1967.
18. Bellazzi R. et. al., "Genomica e proteomica computazionale", Bologna, Pàtron (Italy, 2007).
19. <http://www.dei.unipd.it/~dicamill/teaching/courses/bpg/lucidi>
20. http://en.wikipedia.org/wiki/Self-organizing_map
21. Haykin, Simon, "9. Self-organizing maps", "Neural networks - A comprehensive foundation", (2nd ed.), Prentice-Hall, 1999.
22. S. C. Johnson, "Hierarchical Clustering Schemes", *Psychometrika*, 2:241-254, 1967.
23. Kanti Mardia et. al., "Multivariate Analysis", Academic Press, 1979.
24. David J. Ketchen, Christopher L. Shook, "The application of cluster analysis in strategic management research: an analysis and critique", *Strategic Management Journal*, 17 (6), 441-458, 1996.
25. Cyril Goutte, Peter Toft, Egill Rostrup, Finn Årup Nielsen, Lars Kai Hansen, "On Clustering fMRI Time Series", *NeuroImage*, 9 (3), 298-310, March 1999.
26. <http://www.cybergrat.eu/>