Research Internship (PRe)

Grégoire Bussière

May 2019 - July 2019

Confidentiality Notice

Non-confidential report and publishable on Internet

A simple model for estimating characteristics of event-related potentials

ENSTA Paris tutor: Laurent Bourgeois

UC Irvine tutor: Joachim Vandekerckhove

Acknowledgments

I would like to thank my advisor Joachim Vandekerckhove for the opportunity of joining his lab during this summer internship. I enjoyed working this project thanks to his help and guidance in my discovery of research. This internship made me discover mathematics for psychology and gave me motivation to go further in this very interesting field.

I would also like to thank Michael D. Nunez for giving us access to his data and for answering all my questions about his previous work while away from the lab.

Contents

Introduction

Abstract

This research project is the very beginning of a much wider project that aims to link a cognitive model with a neuro-electric one. The project has been inspired by the paper from Wu et al. [1] but with a different methodology. My job will focus on the neuro-electric model that I will create and adapt to electroencephalogram (EEG) data in order to catch as much information as possible on it and study model parameters that can be estimated on the data. There are three main steps in the work conducted during this internship:

1. Develop a neuro-electric model that is adapted to the EEG data
2. Find an optimization method to conduct curve fitting on the data
3. Implement the model in a Bayesian framework

The model has been modified several times and adapted little by little to be as close as possible to the real data. The research of an efficient optimization method is conducted to prove abilities of the model to fit the data before developing an MCMC (Markov Chain Monte Carlo) algorithm to obtain distributions of the model parameters instead of most likely value.
All the codes can be found and used freely to the Github:
`https://github.com/gregoirebussiere/EEG_Model`

Definitions

Electroencephalogram (EEG) An electroencephalogram is the result of electroencephalography which is an electrophysiological monitoring method to record electrical activity of the brain. It is typically noninvasive, with the electrodes placed along the scalp. EEG measures voltage fluctuations within the neurons of the brain. Diagnostic applications generally focus either on event-related potentials or on the spectrum of the EEG.[2]

Electrodes An electrode is a electric receiver that will record brain electric waves activity. For a classic EEG, 128 of them are used and placed all over the scalp at precise locations that are conventional. They will detect the time evolution as well as the

spatial evolution thanks to this particular disposition. Indeed, some brain responses only occur in a well-defined area. Therefore, depending on the phenomena considered, study electrodes will be selected to focus on a specific area.

Event-related potential (ERP) An event-related potential is the brain electrical answer to a specific stimulus (whether it be motor, cognitive or sensory). Depending on the stimulus, and the brain region of measurement, ERP can be identified as they present a stable pattern that can be recognized in the EEG.

Objectives

In order to understand better the whole project and how this report is included in it, here are the main objectives:

Implement an process that can generate EEG data

Before estimating parameters on real EEG data for which the values of parameters are not known, an algorithm able to generate a random data set must be implemented to run validation tests of the chosen algorithm.

Program an algorithm for parameters estimation

This objective boils down to the implementation of a curve-fitting algorithm (Gauss-Newton, Levenberg-Marquardt, Nelder-Mead simplex...)

Apply the selected algorithm to real EEG data

There is a gap between handmade data and real collected brain data. Once the algorithm is efficient for generated data, it must be tested on real data and if need be, adapted.

Implement this process in a Bayesian framework

Basic statistical inference can be very limited when it comes to understanding the distribution of the model parameters. Indeed, Bayesian inference is very flexible and can catch underlying characteristics.

Joint the neuro-electric model with the cognitive model

The data used for this study has been gathered during a decision making experience and the final objective is to join the drift model for decision making and the model developed in this study thanks to this paper from Joachim Vandekerckhove [3].

1 Model

1.1 General model

An electroencephalogram is the sum of all electrical currents that are measured by the electrode. Thus, an EEG is composed of many simultaneously ongoing brain processes. Therefore, when experiments are carried out, it can be difficult to identify and discriminate a particular ERP as random other brain processes occur in parallel. In order to identify a particular ERP, the general model considers that an EEG is the sum of ERPs and a background white noise that represents all other simultaneous brain activities. It is also crucial to notice that the model depends on the conditions of the trial, the participant, the location of the electrode and the trial itself (different noise each time). Therefore, the EEG will depend on these four factors that have a significant influence on the measured EEG. The general model of the signal measured by an electrode is:

$$\mathbf{y}^{i}_{(l,r,p)}(n) = \left[\sum_{j=1}^{J} a^{j}_{(l,p)} \times \mathbf{c}^{i,j}_{(p)} \times f^{j}\left(n + \tau^{j}_{(l,p)}\right)\right] + \mathbf{b}_{(l,r,p)}(n) \quad \text{for i} \in S$$

where

S	is the set of electrodes
l, r, p	stand for conditions,trials and participants respectively
n	is the time sampling
J	is the total overall ERP components considered
$a^{j}_{(l,p)}$	is the amplitude of the ERP
$c^{i,j}_{(p)}$	is the spatial pattern of the ERP
f^{j}	is the j-th ERP stereotyped contribution to the EEG
$\tau^{j}_{(l,p)}$	is the latency of the ERP relatively to its stereotyped time course
$\mathbf{b}_{(l,r,p)}$	is the background white noise (a null mean and a variance σ^2)

This model considers all factors with an influence on the collected data. Its complexity depends on the number of ERP considered J, as well as the functional forms of the ERPs f^{j}, and it can quickly reach a large number of parameters. In this study, the focus will be on three ERPs whose parameters are willing to be estimated.

1.2 Simplified model

In our case, we only consider 3 ERP components (N200,P300 and Readiness Potential) to simplify the model. These 3 ERPs are observed during very specific brain activities:

N200 is a negative peak observed around 200 ms after the stimulus and preceding the motor answer. It is then supposed to be linked to the identification and processing of the stimulus. It is usually observed when performing an oddball paradigm (display of an unexpected stimulus).

P300 is a positive peak observed around 300/400 ms after a stimulus but can vary from 250 to 900 ms. It is one of the most studied information processing ERP as its identification is relatively easy and is generally associated with the study of the N200. [4]

Readiness Potential (RP) or Bereitschaftspotential is a slow increasing negative potential preceding a hand or foot movement and its amplitude varies according to the subject willingness to perform the movement. The extremum is generally reached around 750 ms after the stimulus.[5]

Only these three ERPs are considered for this study. To begin with, the focus will only be on one electrode. Therefore, no spatial influence will be studied for now and then the term $c^{i,j}$ will be omitted. The aim is only to try recovering parameters from artificial data before trying to apply it to the real brain EEG. Then, our unitary model is:

$$\mathbf{y}(n) = \Phi^{N200}(n) + \Phi^{P300}(n) + \Phi^{RP}(n) \ + \mathbf{b}(n) \text{ (Figure 1)}$$

where

$$\begin{aligned} \Phi^{N200}(n) &= a_{N200} \times f^{N200}(n) \\ \Phi^{P300}(n) &= a_{P300} \times f^{P300}(n) \\ \Phi^{RP}(n) &= a_{RP} \times f^{RP}(n) \end{aligned}$$

The functional forms of the ERPs f^j are likely to change during the project as there are no conventional expressions. The first ones to be considered are:

$$\begin{aligned} f^{N200}(n) &= -\tfrac{n-\tau_{N200}}{\sigma_{N200}} \times \exp\big(-\tfrac{1}{2}(\tfrac{n-\tau_{N200}}{\sigma_{N200}})^2\big) \\ f^{P300}(n) &= \exp\big(-\tfrac{1}{2}(\tfrac{n-\tau_{P300}}{\sigma_{P300}})^2\big) \\ f^{RP}(n) &= -\mathbf{1}_{n<\tau_{RP}+\sigma_{RP}} \times \exp\big(-\tfrac{1}{2}\ln(1+\tfrac{\tau_{RP}-n}{\sigma_{RP}})^2\big) \end{aligned}$$

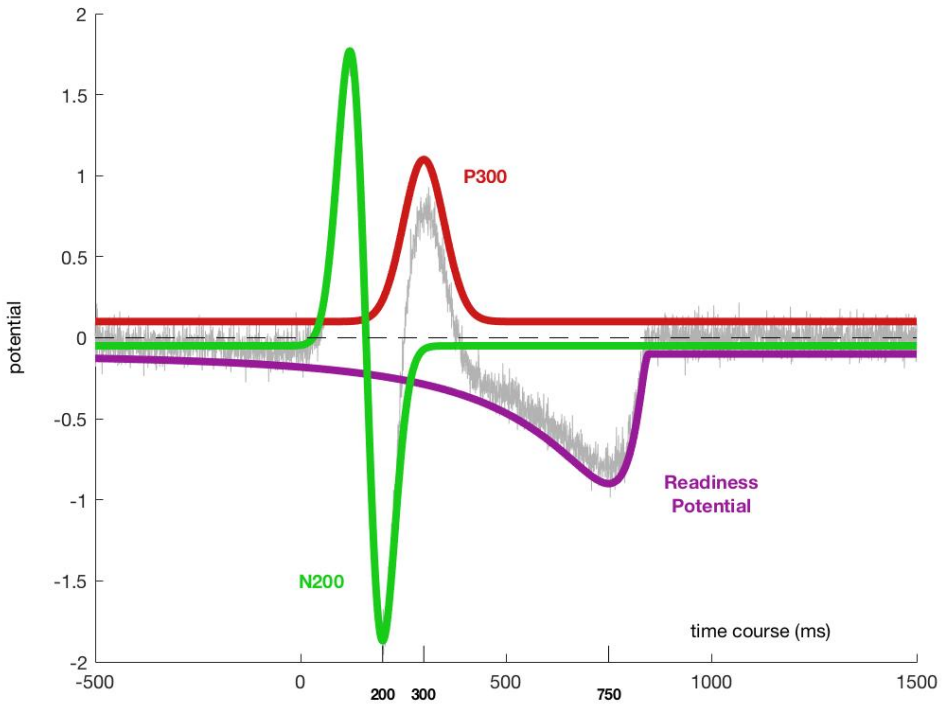

Figure 1: Stereotyped EEG

Therefore, one can recognize the first derivative of a Gaussian for the functional form of the N200, a Gaussian for the P300, and an inverted log-normal for the RP. Each ERP is defined by three parameters, one amplitude parameter a_j, a time parameter τ_j, and a scale parameter σ_j.

Thus, the model is composed of 9 parameters to be estimated on EEG records. Several optimization algorithms will be tested before picking the most efficient one and in order to evaluate them, a script needs to be written to generate training data on which tests will be carried out before applying them to real data.

1.3 Data generation

The aim of this section is to generate artificial data randomly around objective values that are to be recovered after several evaluations. A classic assumption is that the parameters are

distributed according to a normal distribution. Therefore, an arbitrary choice of objective values is selected and training parameters are generated around these objective values. In the present case, the selected vector of values is (same values as the ones in Figure 1) :

$$\boldsymbol{\theta_0} = \big(\underbrace{3\ 200\ 80}_{N200}\ \underbrace{1\ 300\ 50}_{P300}\ \underbrace{0.8\ 750\ 100}_{RP}\big)$$

The standard deviation chosen is 10% of the values and therefore, training data is generated according to $\mathcal{N}\left(\boldsymbol{\theta_0}, {\boldsymbol{\Sigma_0}}^2 = (\frac{\boldsymbol{\theta_0}}{10})^2\right)$. The parameters are considered independent, then for an easier readability $\boldsymbol{\Sigma_0}$ is written as a vector of the variance of each parameters. A Gaussian noise is also added to obtain a more realistic EEG.

Once the data generated, the aim is to recover parameters for each training EEG and recover θ_0 by maximum of likelihood : $\hat{\boldsymbol{\theta_0}} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\theta}^i$ with $\boldsymbol{\theta}^i$ the estimated vector of parameters on trial i.

2 Optimization methods to estimate the parameters

2.1 Objective function

In order to estimate the parameters of our EEG model on the observed data, the first idea that comes to mind is to solve a non-linear least-squares curve fitting problem. The function to minimize is :

$$F(\boldsymbol{\theta}) = \frac{1}{2}\sum_{n=1}^{N}\left[y(n) - f\left(n, \boldsymbol{\theta}\right)\right]^2 = \frac{1}{2}\sum_{n=1}^{N} r_n^2$$

with
- y the data
- f the model approximation

2.2 Gauss-Newton Algorithm

The first optimization method considered is the Gauss-Newton algorithm. This algorithm is an approximation of Newton's method that proposes a recurrence relation :

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \left(\mathbf{J}_F^\top \mathbf{J}_F\right)^{-1}\mathbf{J}_F^\top \mathbf{r}\left(\boldsymbol{\theta}^{(k)}\right)$$

with $\mathbf{J}_F$ the Jacobian and $\mathbf{r}\left(\boldsymbol{\theta}^{(k)}\right)$ the residuals instead of Newton's method's recurrence relation :

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \mathbf{H}^{-1}\mathbf{g}$$

with $\mathbf{H}$ the Hessian and $\mathbf{g}$ the gradient of F. The difference between the two methods is the approximation $\mathbf{H} \approx 2\mathbf{J}_F^\top \mathbf{J}_F$ in the Gauss-Newton method that is simpler and avoid to calculate the Hessian which is very demanding in application.
The results obtained with this method were not satisfying as the algorithm did not converge. Around the minimum, convergence was obtained but when starting away from it, only divergence was observed. Even the implementation of a step verifying the Wolfe conditions did not solve the problem of convergence.

2.3 Levenberg-Marquardt Algorithm

The second algorithm considered was the Levenberg-Marquardt algorithm. Its main asset is flexibility and stability compared to the Gauss-Newton method. Indeed, if an iteration is effective, the method will be close to Gauss-Newton algorithm, but it can also be close to the gradient method thanks to the addition of a parameter λ. The recurrence relation used in this algorithm is [6]:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - (\mathbf{H} + \lambda \operatorname{diag}[\mathbf{H}])^{-1}\mathbf{g}$$

The value of λ is updated at each iteration to modulate the size of the step. The update rule is :

1 : Calculate $\boldsymbol{\theta}^1_{temp}$ and $\boldsymbol{\theta}^2_{temp}$ according to the recurrence relation above for the current value of λ and $\frac{\lambda}{\nu}$, $\nu > 1$ ($\nu = 10$ generally) and evaluate the value of $F(\boldsymbol{\theta}^1_{temp})$ and $F(\boldsymbol{\theta}^2_{temp})$.

2 : If $F(\boldsymbol{\theta}^2_{temp}) < F(\boldsymbol{\theta}^{(k)})$, $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^2_{temp}$ and λ is updated to the value $\frac{\lambda}{\nu}$.

3 : If $F(\boldsymbol{\theta}^1_{temp}) < F(\boldsymbol{\theta}^{(k)}) < F(\boldsymbol{\theta}^2_{temp})$, $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^1_{temp}$ and the value of λ is conserved.

4 : Otherwise, multiply λ by ν until you get a lower value of F.

In application, $\mathbf{H}$ is approximated by $2\mathbf{J}_F^\top \mathbf{J}_F$ like in the Gauss-Newton method to reduce calculations. The results with this algorithm were slightly better than the Gauss-Newton's ones but when the first iteration is not close to the minimum, divergence was observed in some cases. One of the possible reasons is that in the jacobian $\mathbf{J}_F$, there are many 0. Even the use of a sparse matrix did not stabilize the iterations. Therefore, instead of using an analytic algorithm with possible divergence, a more heuristic algorithm has been considered.

2.4 Nelder-Mead Simplex

Nelder-Mead simplex or also called downhill simplex method is a commonly used heuristic method for finding the minimum of a function. Even if it can converge to non-stationary points, its efficiency and easy implementation make him a good alternative to traditional methods when the objective function is smooth enough and unimodal. This algorithm uses a simplex, which is a set of $n+1$ vertices in a space of dimension n. The method is [7]:

0 Initialization: Choose an initial simplex (this choice is very important for the efficiency) $\{\boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_{n+1}}\}$ and order it according to the values of F for this vertices. The initial simplex is then:

$$S_0 = \{\boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_{n+1}}\} \text{ with } F(\boldsymbol{\theta_1}) \leq F(\boldsymbol{\theta_2}) \leq \ldots \leq F(\boldsymbol{\theta_{n+1}})$$

1 Termination: Check the termination criteria ($|F(\boldsymbol{\theta_1}) - F(\boldsymbol{\theta_{n+1}})| < \epsilon$, max iterations reached...).

2 Centroid: Calculate the centroid $\boldsymbol{\theta_0}$ of all vertices except x_{n+1}.

3 Reflection: Calculate the reflected point $\boldsymbol{\theta_r} = \boldsymbol{\theta_0} + \alpha(\boldsymbol{\theta_0} - \boldsymbol{\theta_{n+1}}),\ \alpha > 0$.
If $F(\boldsymbol{\theta_1}) \leq F(\boldsymbol{\theta_r}) < F(\boldsymbol{\theta_n})$, then $\boldsymbol{\theta_{n+1}} = \boldsymbol{\theta_r}$ and go to step **7**.

4 Expansion: If $F(\boldsymbol{\theta_r}) < F(\boldsymbol{\theta_1})$, then calculate $\boldsymbol{\theta_e} = \boldsymbol{\theta_0} + \gamma(\boldsymbol{\theta_r} - \boldsymbol{\theta_0}),\ \gamma > 1$.
If, $F(\boldsymbol{\theta_e}) < F(\boldsymbol{\theta_r})$, then $\boldsymbol{\theta_{n+1}} = \boldsymbol{\theta_e}$ and go to step **7**.
Otherwise, $\boldsymbol{\theta_{n+1}} = \boldsymbol{\theta_r}$ and go to step **7**.

5 Contraction: Here, $F(\boldsymbol{\theta_r}) \leq F(\boldsymbol{\theta_n})$. Calculate the contracted point
$\boldsymbol{\theta_c} = \boldsymbol{\theta_0} + \rho(\boldsymbol{\theta_{n+1}} - \boldsymbol{\theta_0}),\ 0 < \rho < 0.5$.
If $F(\boldsymbol{\theta_c}) < F(\boldsymbol{\theta_{n+1}})$, then $\boldsymbol{\theta_{n+1}} = \boldsymbol{\theta_c}$ and go to step **7**.

6 Shrink: Replace all $\boldsymbol{\theta_k}$ with $\boldsymbol{\theta_{n+1}} = \boldsymbol{\theta_1} + \sigma(\boldsymbol{\theta_k} - \boldsymbol{\theta_1}),\ 0 < \sigma < 1$ except for $\boldsymbol{\theta_1}$ and go to step **7**.

7 Ordering Order the simplex.

There are typical values for the parameters $\alpha, \gamma, \rho, \sigma$ $(1, 2, 0.5, 0.5$ respectively) but in order to obtain the most efficient algorithm, the value of alpha will be chosen after several tests on a grid. To do so, data is generated and a grid is created in $[0.8; 0.999]$. For each value

of the grid, the algorithm is run and $F(\boldsymbol{\theta_{opt}})$ is calculated. The result is a graph like Figure 2:

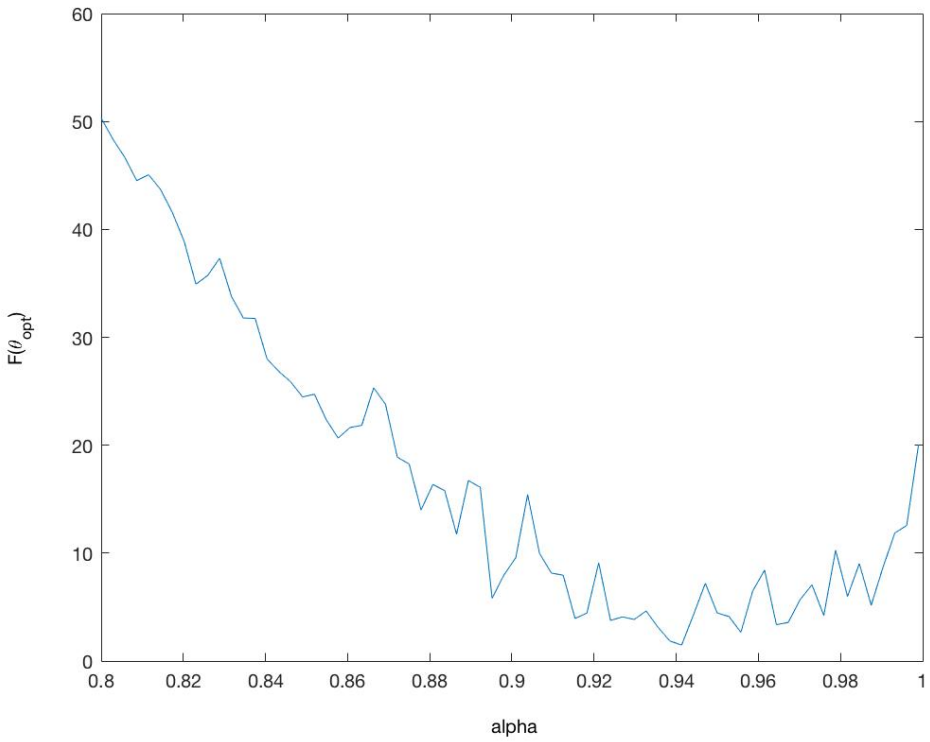

Figure 2: Selection of the best value for α

For each iteration, the best α is recorded and after repeating this process many times, the final α selected for the further study is the mean of the values recovered.

With this value of α, convergence is reached after approximately 350 iterations and a time of less than 0.2 second which is efficient enough for this study. The decrease in the values of F can be observed in Figure 3.

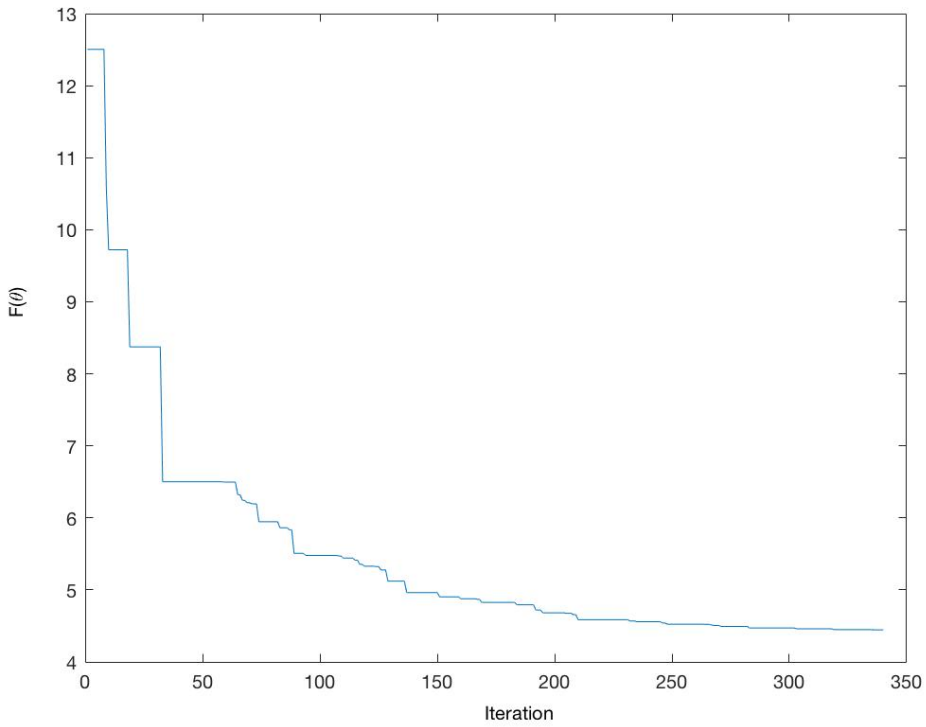

Figure 3: Values of F during the Nelder-Mead Simplex

The Nelder-Mead Simplex is the only method that give stable results for each trial, and is the algorithm considered for the training part. Some results on generated EEGs can be observed on Figure 4, the blue lines being the data and the yellow line the estimation after the optimization process:

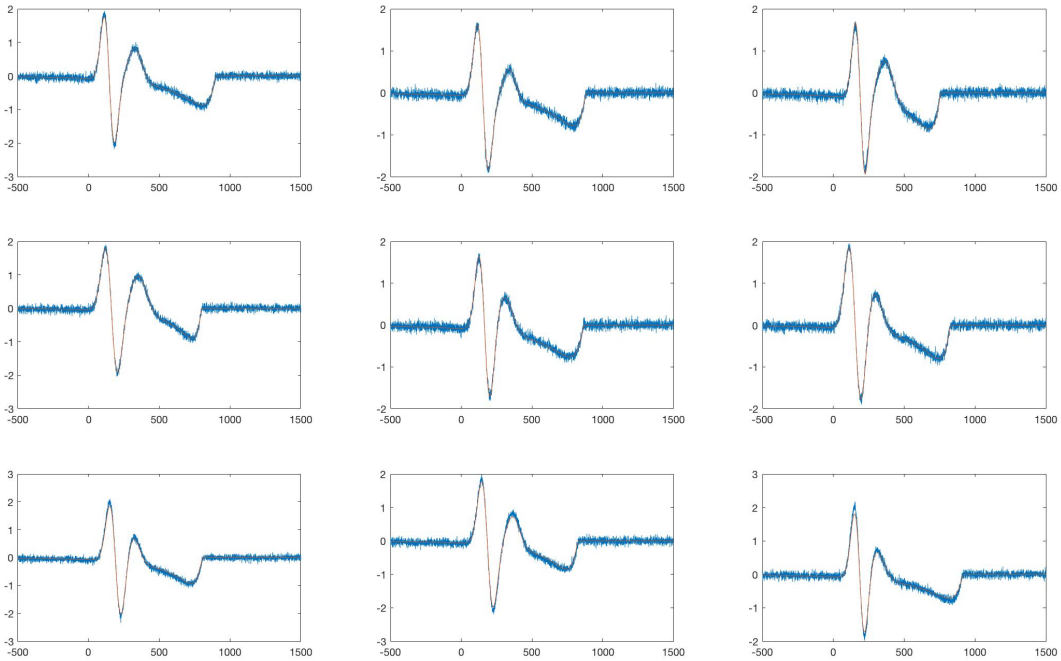

Figure 4: Approximation of some training data after optimization

The efficiency of this optimization method is clear on these plots, the noise does not affect the estimation of the parameters that are recovered very precisely. A computation of the relative error $\frac{\Delta \boldsymbol{\theta}}{\boldsymbol{\theta}}$ for each parameter and each EEG generated gives the following plots on Figure 5:

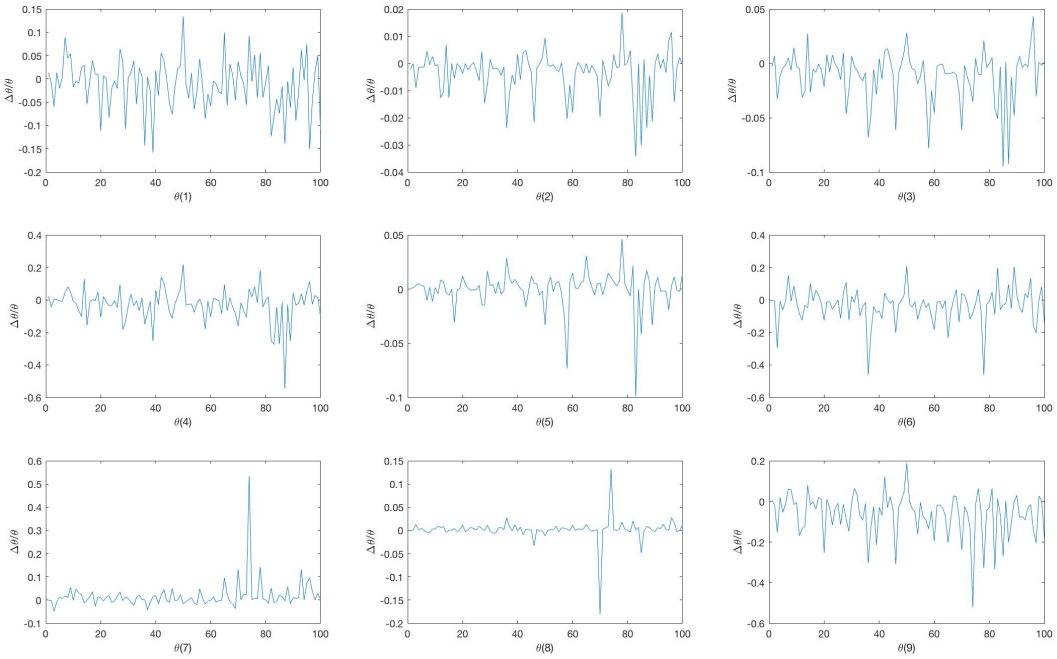

Figure 5: Relative errors obtained for the estimations of $\boldsymbol{\theta}$

The plots of Figure 5 show that the error committed remains low in the majority of the trials. However, it can be seen that the errors do not look random (for instance $\boldsymbol{\theta}(2)$ has a majority of negative errors). Let's check if these errors can be assimilated to a Gaussian noise by displaying the QQ-plots:

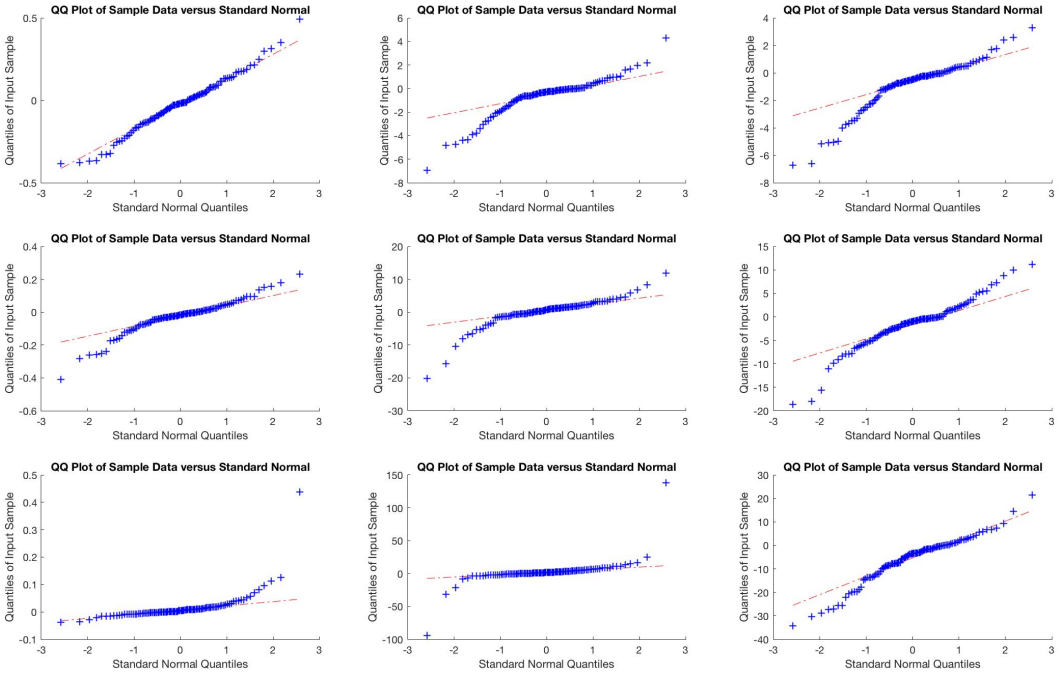

Figure 6: QQ-plots for the residuals of the estimations of each parameter

These QQ-plots can't corroborate the assumption of Gaussian residuals, especially for $\boldsymbol{\theta}(2)$, $\boldsymbol{\theta}(3)$, $\boldsymbol{\theta}(5)$ and $\boldsymbol{\theta}(6)$. This phenomenon has been identified since the first version of the optimization method, but even with modifications in its structure, the displayed result is the best obtained. This is a bit concerning as this can lead to a systematic error during the estimation of the parameters. However, the absolute value of the error remains small enough not to consider this phenomenon significant enough to alter the final results too much (see the final results in **3.5**. But being aware of this can be useful and can be a lead to improve the efficiency and precision of this method.

2.5 Recovery of the parameters

Once the algorithm ready and efficient on generated data, the next step is to try recovering the initial parameters by maximum of likelihood estimation (MLE):

$$\boldsymbol{\theta_0} = \begin{pmatrix} 3 & 200 & 80 & 1 & 300 & 50 & 0.8 & 750 & 100 \end{pmatrix}$$

The estimator of maximum likelihood for $\boldsymbol{\theta_0}$ is $\hat{\boldsymbol{\theta_0}}(j) = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{\theta_0}^i(j)$ with $\boldsymbol{\theta_0}^i$ the estimated vector of parameters on trial i.
This estimator has been calculated for 100 EEGs generated and the values obtained for $\hat{\boldsymbol{\theta_0}}$ is :

$$\hat{\boldsymbol{\theta_0}} = \big(3.07\ 196\ 80.3\ 1.03\ 295\ 50.8\ 0.776\ 744\ 105\big)$$

The relative error for the estimation is then:

$$\frac{\boldsymbol{\theta_0} - \hat{\boldsymbol{\theta_0}}}{\boldsymbol{\theta_0}} = \big(-0.02\ 0.02\ \ -0.004\ \ -0.03\ 0.01\ \ -0.02\ 0.03\ 0.008\ \ -0.05\big)$$

which represent between 0.4% and 5% of error depending on the parameter considered when recovering the initial parameters. These values could have been lower if more EEGs were evaluated but the ones obtained are low enough to confirm the efficiency of this method for evaluating the true values of the parameters $\boldsymbol{\theta_0}$.
The variance of the Gaussian distribution has also been recovered by maximum of likelihood $\big(\hat{\boldsymbol{\Sigma_0}}(j)^2 = \sum_{i=1}^{N}(\boldsymbol{\theta}(j)^i - \hat{\boldsymbol{\theta}}(j))^2\big)$ and the results were also close to the initial values ($\approx 10\%$ of error).

MATLAB has a built-in function *fminsearch* that uses the Nelder-Mead simplex as well to find the minimum of a function. The results of comparing the built-in algorithm with the hand-made one resulted in an advantage for the built-in function (Figure 7 and 8). Therefore, even if the parameters can't be modified and the built-in algorithm is like a black box, it will be used as efficiency is the main objective.

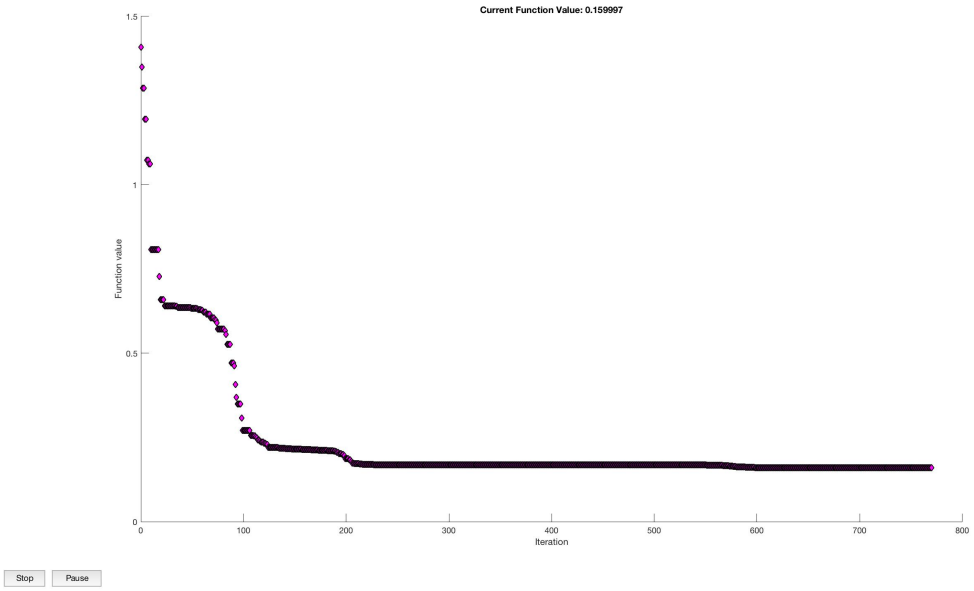

Figure 7: Efficiency of the built-in function in MATLAB

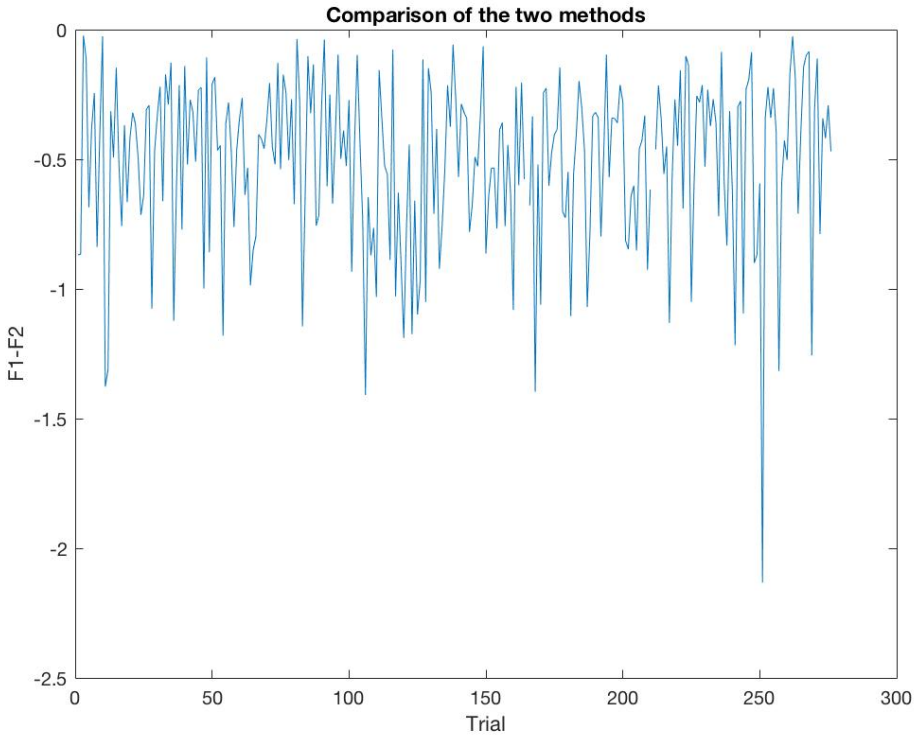

Figure 8: Difference between the values of F obtained by the two algorithms

3 Tests on real data

The real data on which this study was conducted comes from experiments carried out by Michael D. Nunez during his PhD research. All details of how the data was gathered are fully explained in his thesis [8].

From this point, only the N200 will be studied. It was more interesting to focus on a single ERP and carry a full study than doing just a bit on several of them. Therefore, the study of P300 and Readiness Potential will not be presented in this document but they will be studied by next members of the research team.

3.1 Data Cleaning

Before being exploitable, EEG data has to be cleaned as it is composed of much noise coming from several origins such as facial motor activity (e.g. a blink) or background brain activity that entail a poor signal-to-noise ratio (Figure 9). Therefore, some filtering must be applied to increase this ratio. This filtering has not been done in this project as it has been previously done by M. Nunez but it is interesting to understand the choice of filters and how EEG data is processed before being exploitable.

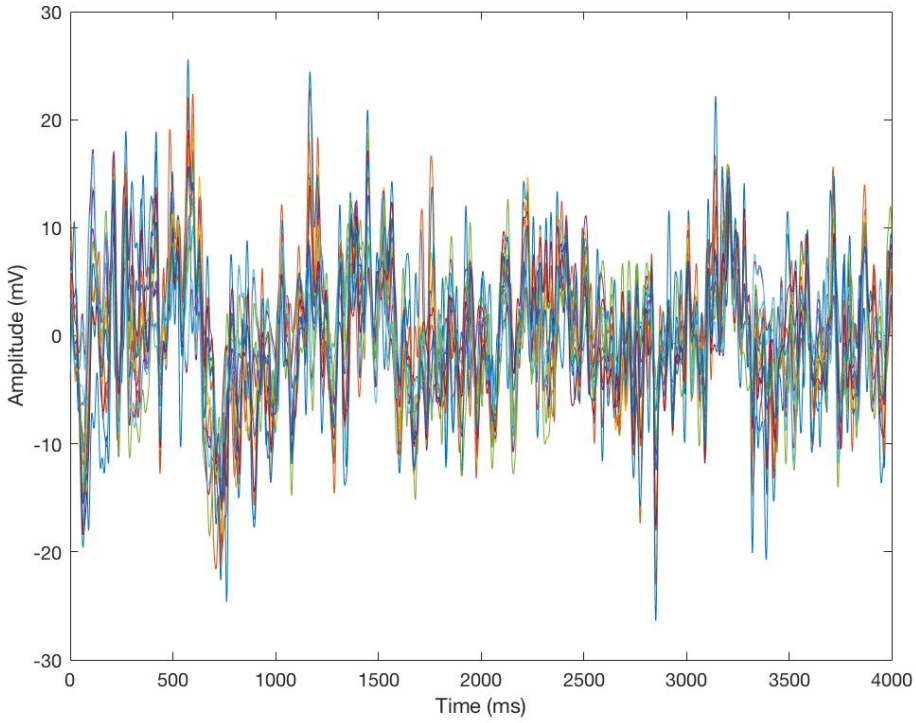

Figure 9: Example of signals obtained by 15 electrodes during a trial

In this subsection, emphasis will be put on the reduction of dimensions in the data by removing the spatial complexity. Indeed, this reduction must be carried out in order to go from 128 channels per trial (one per electrode) to a single channel without losing too much information during the process. The selected method is principal component analysis (PCA) as it is quite simple but effective to catch as much information about the data as possible

only with the few first components.

Here is a brief reminder of how PCA works:
In order to reduce the dimension of the data, each electrode will be considered as a variable of the data. The aim is to find a few axes that are linear combinations of the variables that represent the best the data, i.e. the axes that conserve the best the variance of the data. The data must first be centered and normalized before calculating the covariance matrix. Then, the eigenvalues λ_i and eigenvectors v_i, $i \in \{1, \dots, 128\}$ are calculated and ordered by the proportion of variance each eigenvalue represents, i.e. ordered according to the value $p_i = \frac{\lambda_i}{\sum \lambda_i}$. On the data, the first component explains between 68% and 79% of the variance depending on the condition selected. Therefore, as this value is reasonably high, and as the N200 peak is very visible when projecting on this first component, for each trial $\mathbf{k}$, the data $\mathbf{Y_k} = \begin{pmatrix} y_k^{1,1} & \dots & y_k^{1,p} \\ \vdots & & \vdots \\ y_k^{n,1} & \dots & y_k^{n,p} \end{pmatrix}$ with n the number of samples for the trial and p the number of electrodes, will be processed in the following linear transformation:

$$\mathbf{X_k} = \mathbf{Y_k}\mathbf{V}$$

with $\mathbf{V} = \begin{pmatrix} v_1^1 \\ \vdots \\ v_1^p \end{pmatrix} = v_1$ a unit eigenvector associated to the largest eigenvalue λ_1.

This data simplification was efficient, especially when the time window considered was the one in which the N200 is likely to occur. However, the PCA could only been done on the raw data as we couldn't access the data after the first filters and as M. Nunez conducted approximately the same method for reducing the dimension, his final data was decided to be the one selected for the study. It would be interesting to conduct the same filtering as he did but with a N200-window focus for dimension reduction and compare the signal obtained to his, as we can expect more adapted weights to highlight the N200.

3.2 Model and objective function modifications

Once the data filtered and cleaned, its dimension for each trial $\mathbf{k}$ is $n \times 1$: $\mathbf{Y_k} = \begin{pmatrix} y_k^1 \\ \vdots \\ y_k^n \end{pmatrix}$ (the notation $\mathbf{Y_k}$ is kept for better comprehension).

After a quick look on the shape of the N200 peak, one can notice that the initial functional form for the peak (derivative of a Gaussian) is not adapted to the data (Figure 10) and the results obtained were really unsuccessful. Moreover, brain activity is very likely to change dramatically even for the same experiment in the same conditions consecutively which makes curve fitting harder and the first model considered obsolete. The model needed more adaptability and needed to be able to fit diverse forms of peaks.

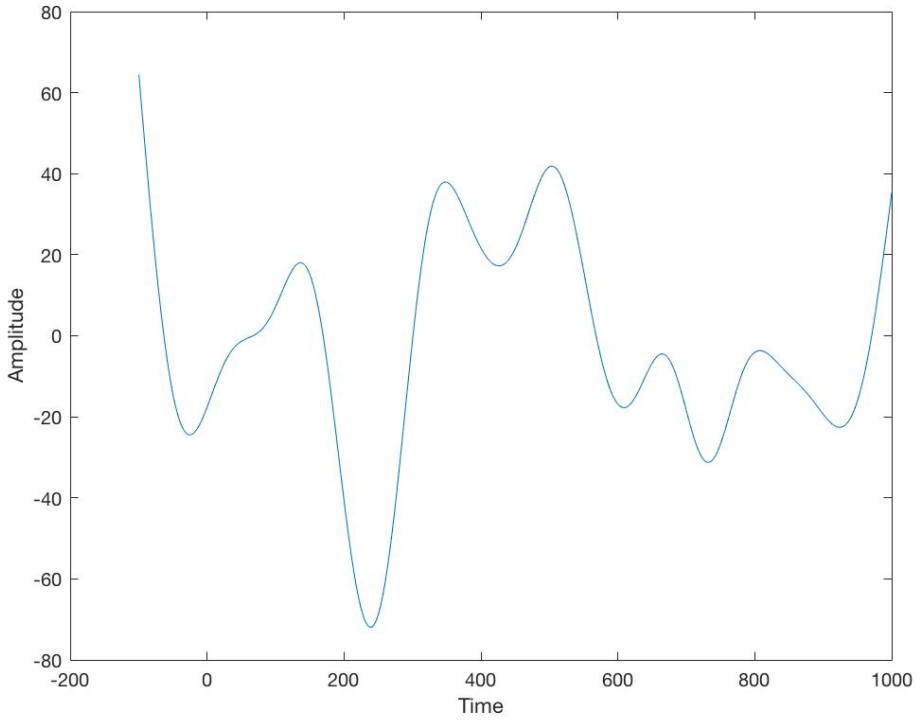

Figure 10: Example of signal obtained on clean data

Then, the model of the N200 had to be changed. The shape around the N200 looks like the

second derivative of a Gaussian and this functional form has been tried. The new model was:

$$f^{N200}(n) = a_{N200} \times \left(\left(\frac{n-\tau_{N200}}{\sigma^2_{N200}} \right)^2 - \frac{1}{\sigma^2_{N200}} \right) \times \exp\left(-\frac{1}{2} \left(\frac{n-\tau_{N200}}{\sigma_{N200}} \right)^2 \right)$$

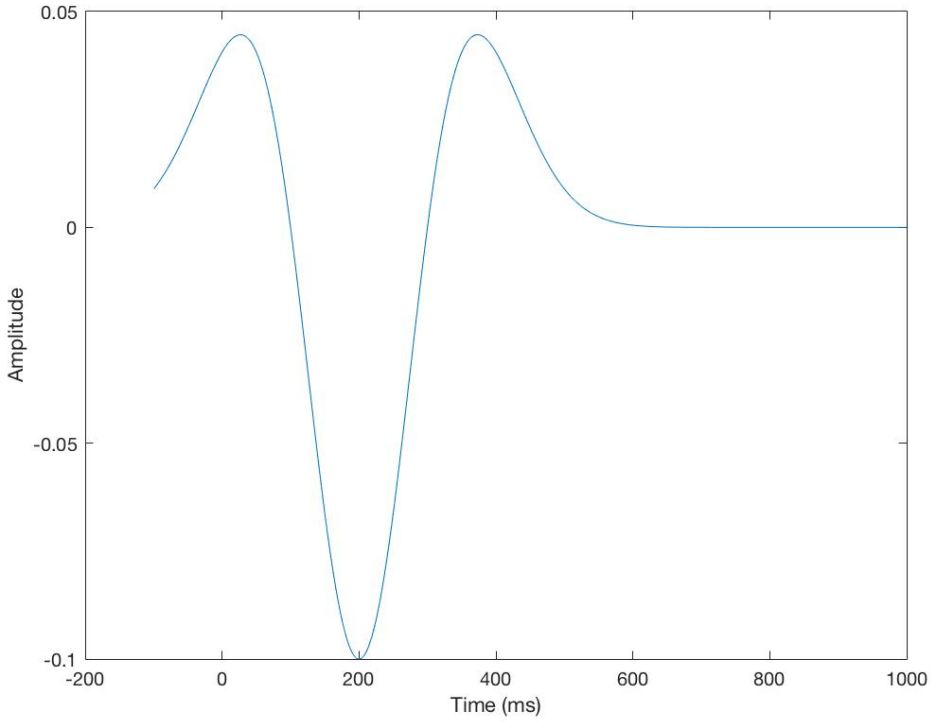

Figure 11: Example of curve obtained with a second derivative of a Gaussian

It is clearly closer to the real shape of the peak but this functional form was not enough to fit all types of peaks encountered. Indeed, N200 peaks are not generally symmetric and this model is. Then, to add more flexibility, the functional form has been divided in two parts, one before the minimum and one after. These two parts are linked by a constraint on the value at the minimum. Also, a parameter was added for the offset that can occur in some EEGs even if during the preprocessing noise has been mostly cancelled. Finally, the selected model is:

$$f^{N200}(n) =$$
$$\begin{cases} a_{N200,1} \times \left(\left(\frac{n-\tau_{N200}}{\sigma^2_{N200,1}} \right)^2 - \frac{1}{\sigma^2_{N200,1}} \right) \times \exp\left(-\frac{1}{2} \left(\frac{n-\tau_{N200}}{\sigma_{N200,1}} \right)^2 \right) + \epsilon_{N200} & if\ n \leq \tau_{N200} \\ a_{N200,2} \times \left(\left(\frac{n-\tau_{N200}}{\sigma^2_{N200,2}} \right)^2 - \frac{1}{\sigma^2_{N200,2}} \right) \times \exp\left(-\frac{1}{2} \left(\frac{n-\tau_{N200}}{\sigma_{N200,2}} \right)^2 \right) + \delta_{N200} & \text{if } n > \tau_{N200} \end{cases}$$

with $\delta_{N200} = \epsilon_{N200} - \frac{a_{N200,1}}{\sigma^2_{N200,1}} + \frac{a_{N200,2}}{\sigma^2_{N200,2}}$

Therefore, the model is composed of 6 parameters:

$$\boldsymbol{\theta} = \begin{pmatrix} a_{N200,1} & \tau_{N200} & \sigma_{N200,1} & \epsilon_{N200} & a_{N200,2} & \sigma_{N200,2} \end{pmatrix}$$

The term δ_{N200} is added for linking the two parts at time $t = \tau_{N200}$. Indeed, the left part has a value of $-\frac{a_{N200,1}}{\sigma^2_{N200,1}} + \epsilon_{N200}$ at this point and the right part has to have the same value at this point to ensure continuity of the model. It is now possible to have an asymmetric functional form (Figure 12) that will be able to fit complex N200 peaks.

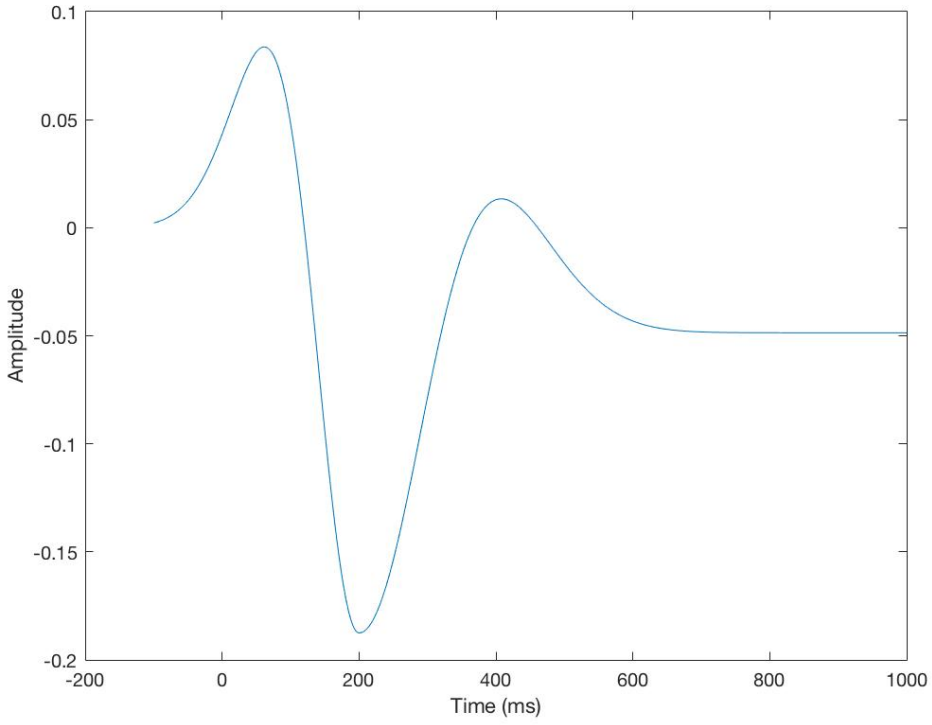

Figure 12: Example of curve obtained with the final model

Once this model ready, a change on the objective function had to be done. Indeed, real data is not just a negative peak around 200 ms, but presents as well others peaks and fluctuation. Therefore, the objective function F would get huge values because of the rest of the signal. It could alter the convergence of the optimization method and therefore a restrained window had to be chosen in order not to consider points outside the N200 peak. To do so, the window chosen was between the extrema of the functional form, i.e. in the peak. The maxima located on each side of the peak were calculated by solving the equation $\frac{\partial f_{N200}}{\partial n} = 0$. This equation is a third order polynomial and in order to solve it, the Tschirnhaus method and the Cardan one were used. The three solutions obtained were:

$$\left(\tau_{N200} - \sqrt{3}\,\sigma_{N200} \quad \tau_{N200} \quad \tau_{N200} + \sqrt{3}\,\sigma_{N200}\right)$$

Then, the function F had as expression:

$$F(\boldsymbol{\theta}) = \frac{1}{2}\sum_{n=1}^{N}\left[y(n) - f\left(n, \boldsymbol{\theta}\right)\right]^2 \mathbf{1}_{n\in[\tau_{N200}-\sqrt{3}\sigma_{N200};\ \tau_{N200}+\sqrt{3}\sigma_{N200}]}$$

After the first tests with the Nelder-Mead simplex, the results were quite good but on several trials, the position of the minimum was not well estimated. It is due to the fact that the function to optimize does not make a difference between the points inside the window. However, one main point of the study is to estimate correctly the timing of the N200. Therefore, it is needed to give a larger weight to the points near the minimum of the peak. Moreover, with the previous function F, the window varies from one estimation to the other. This is why some penalization weights were implemented. The weights are values on a Gaussian distribution and the parameters chosen so that the weights are close to 0 before 100ms and after 400ms. The values retained are a mean of τ_{N200} and a standard deviation of 100. Finally, the function F is calculated thanks to:

$$F(\boldsymbol{\theta}) = \frac{1}{2}\sum_{n=1}^{N}\left[\left(y(n) - f\left(n, \boldsymbol{\theta}\right)\right)^2 w(n, \boldsymbol{\theta})\right] = \frac{1}{2}\sum_{n=1}^{N} R_n$$
$$\text{with } w(n, \boldsymbol{\theta}) = \exp\left(-\frac{1}{2}\left(\frac{n-\tau_{N200}}{100}\right)^2\right)$$

3.3 Estimation of the parameters

With this function F, it is more likely that the minimum is well estimated as the points near the time τ_{N200} have a much larger weight than those further. With this method, the results obtained were pretty good even if we observed some cases for which the simplex did not managed to find a good solution (Figure 13). It was generally due to a signal that did

not look like the others, maybe because of a particular brain activity for this trial. After writing a script able to analyze if the solution given can be considered as good enough, all trials without a satisfactory solution were discarded. That way, it was sure that with the trials left, it would be possible to obtain an efficient solution during the MCMC algorithm.

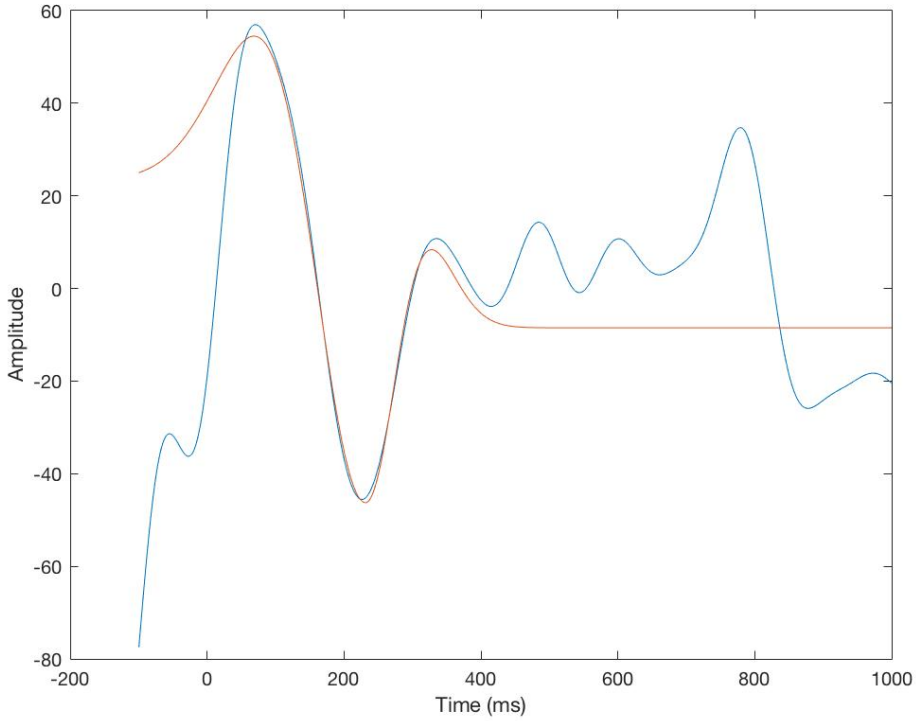

Figure 13: Example of curve fitting obtained with the final model

This method of estimation seems then effective and able to catch and recover parameters that give good curve fitting. Once sure that this method is effective, it can be applied in a Bayesian algorithm and obtain distributions of the parameters instead of most likely values that do not give that much information, especially concerning the uncertainty of the results.

4 Bayesian inference by MCMC

As explained previously, the choice of doing curve fitting instead of calculating the most likely value is to conduct a Bayesian inference thanks to an MCMC algorithm in order to get distributions of the parameters instead of single values. The work done previously permitted to confirm if the model chosen is adapted enough to the real data and now that the model is good enough, we have to implement it in an MCMC algorithm.

4.1 Choice of the MCMC algorithm

There are many Markov Chain Monte Carlo algorithm, and in order to choose the best adapted to this problem, it is important to know how and when the different algorithms are to be used. The book Markov chain Monte Carlo: stochastic simulation for Bayesian inference [9] has been really precious for this study and has given a good overall analysis on this type of algorithms. The retained algorithm is a Metropolis algorithm which is one of the simplest MCMC algorithms. This choice has been made for several reasons: it is easy to implement, very flexible without too much changes to do in case of a model modification (contrary to Gibbs sampling or Hamiltonian Monte-Carlo). Moreover, there was no interest in choosing an asymmetric transition kernel like the Metropolis-Hastings algorithm suggests, therefore the Metropolis one is the right pick to choose.
Without explaining the theory behind this algorithm that would be very long, the most important characteristics of the algorithm will be detailed. Every MCMC algorithm is base on the Bayes theorem which states that for the data y observed:

$$\pi(\boldsymbol{\theta}) = \frac{\mathbb{P}(y|\boldsymbol{\theta})\mathbb{P}(\boldsymbol{\theta})}{\mathbb{P}(y)} = \frac{\mathcal{L}(y,\boldsymbol{\theta})\mathbb{P}(\boldsymbol{\theta})}{\mathbb{P}(y)}$$

where

$\pi(\boldsymbol{\theta})$	is the posterior of $\boldsymbol{\theta}$
$\mathcal{L}(y,\boldsymbol{\theta}) = \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(-\frac{R_n}{2\sigma_n^2}\right)$	is the likelihood of the data
$\mathbb{P}(\boldsymbol{\theta}) = \mathcal{N}(\mu, \Sigma)$	is the prior of $\boldsymbol{\theta}$
$\mathbb{P}(y)$	is the probability of observing the data

This theorem gives the possibility to access information on the real probability distribution of the parameters by modulating a prior with the data observed. Thanks to this theorem and the theory behind Markov chains, it is possible to draw samples from an unknown distribution which is the aim of this study in order to obtain estimated probability distributions of the parameter $\boldsymbol{\theta}$.

The idea behind the Metropolis algorithm is to explore the space of variables following a Markov chain thanks to a symmetric transition kernel $\mathcal{Q}$ and converge in areas where the posterior $\pi(\boldsymbol{\theta})$ is large. The convergence to the real probability distribution is assured by the ergodic theorem.
The Metropolis algorithm is mainly composed of these steps:

0 Initialization: An arbitrary value is given to $\boldsymbol{\theta}$. Then, the log-likelihood $log\mathcal{L}(y, \boldsymbol{\theta})$ and the log-prior $log\mathbb{P}(\boldsymbol{\theta})$ are calculated. Then, the log-posterior can be obtained: $log\pi(\boldsymbol{\theta}) = log\mathcal{L}(y, \boldsymbol{\theta}) + log\mathbb{P}(\boldsymbol{\theta})$

1 New candidate: Draw a new point $\boldsymbol{\phi}$ from the kernel distribution $Q(\boldsymbol{\theta})$. Evaluate its log-posterior and calculate the value $\alpha = log\pi(\boldsymbol{\phi}) - log\pi(\boldsymbol{\theta})$.

2 Acceptation: Draw $u \sim \mathcal{U}(0, 1)$. If $log(u) < \alpha$, $\boldsymbol{\theta} = \boldsymbol{\phi}$.

3 Save: Save the current value of $\boldsymbol{\theta}$ and go back to **1**.

The first values of the chain must not be saved as before a certain number of iterations, the Markov chain is not stationary. This step is called burn-in and the number of iterations will depend on the problem.
The prior has been chosen in order not to restrain too much the values of parameters. Therefore, normal distributions were selected for all parameters with a large standard deviation. Moreover, the assumption of independence of parameters is made even if it can be called into question.
The terms σ_n are for now constant but in a further study, they should be modified.
The kernel $\mathcal{Q}$ has been chosen after several trials in order to explore sufficiently the space of variables but not too much in order to accept quite often the next candidate (objective of 30/40% of acceptation).

The results obtained with this algorithm can be seen on Figure 14. The part in red represents the burn-in period (fluctuations can be observed) and the blue part is the part saved for the analysis. The six first graphs are the six parameters of $\boldsymbol{\theta}$ and the last one is the log-likelihood.

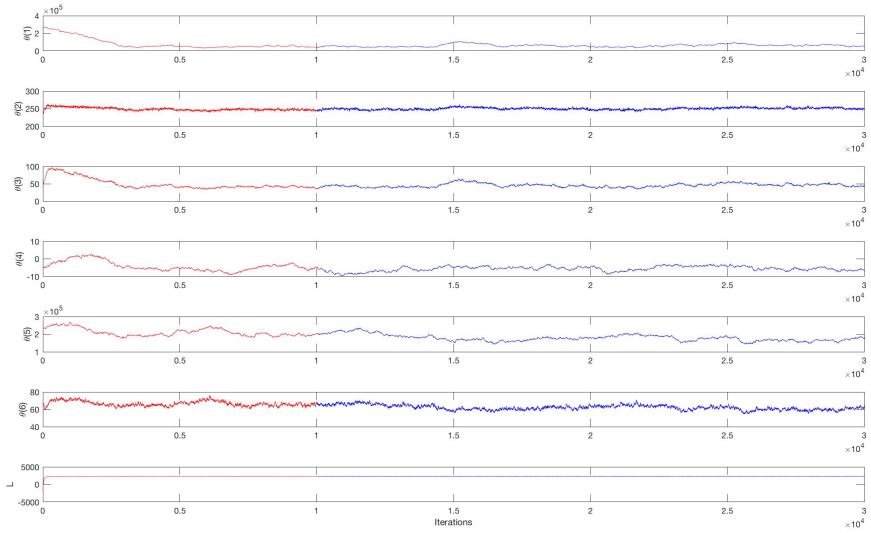

Figure 14: Example of chain obtained with the Metropolis algorithm

4.2 Probability distributions of the parameters

The histogram of the blue part of the chain generated can be plotted and should be close the real distribution of the parameters. As it is a heuristic algorithm, results are different for each chain generated but the general form of the distributions remains the same.

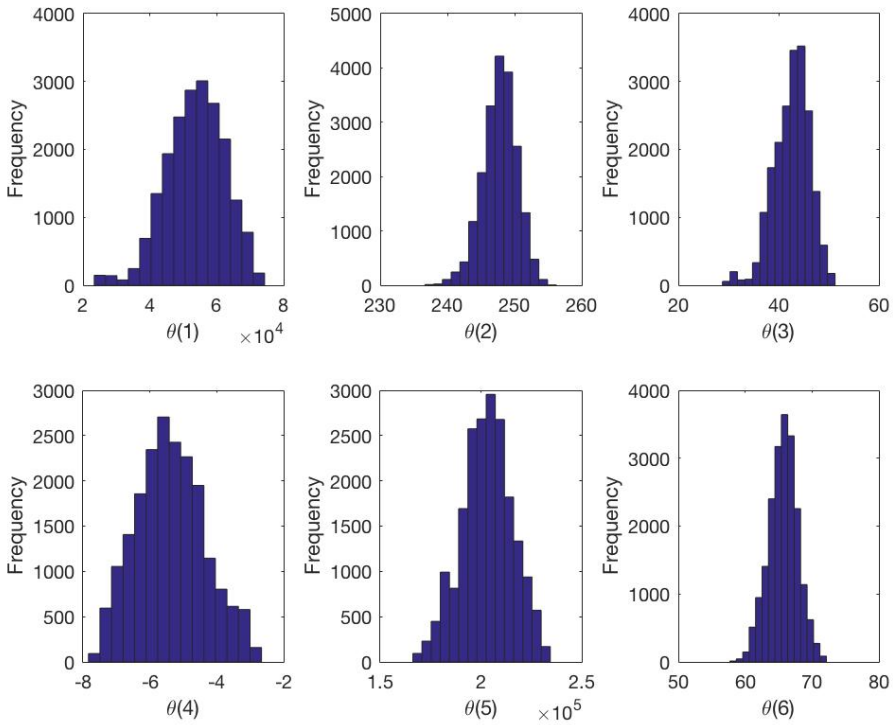

Figure 15: Histograms of the Markov chain for each parameter of $\boldsymbol{\theta}$

The shapes of the histograms look like a Gaussian distribution. Therefore, by fitting a Gaussian distribution on these histograms, a mean and a variance can be measured (Figure 16). With the mean, it is possible to plot the model with these values and compare it to the real data as well as the values obtained with the Nelder-Mead Simplex. This comparison can be seen in the Figure 17. The blue signal is the real data, the orange one is the results of the Nelder-Mead simplex and the yellow curve is the one obtained with the means of the estimated Gaussian on the MCMC histograms.

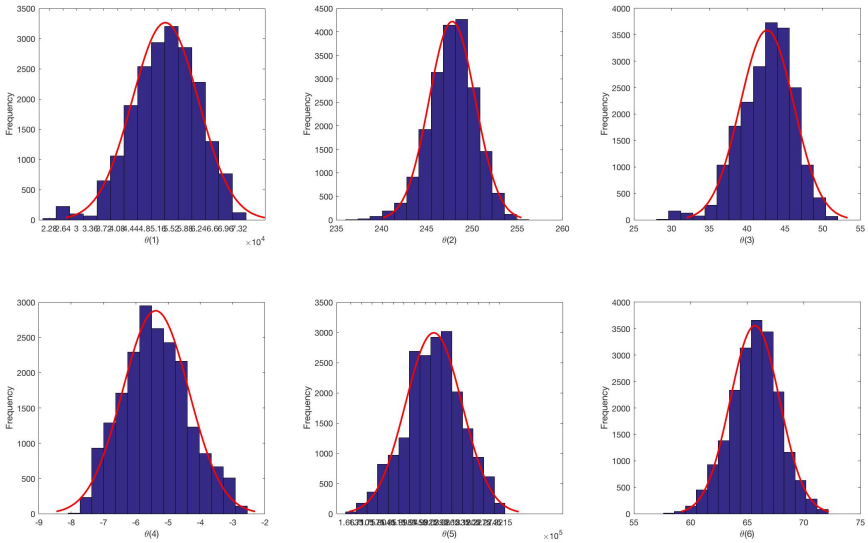

Figure 16: Gaussian distribution fitting on the histograms obtained by MCMC

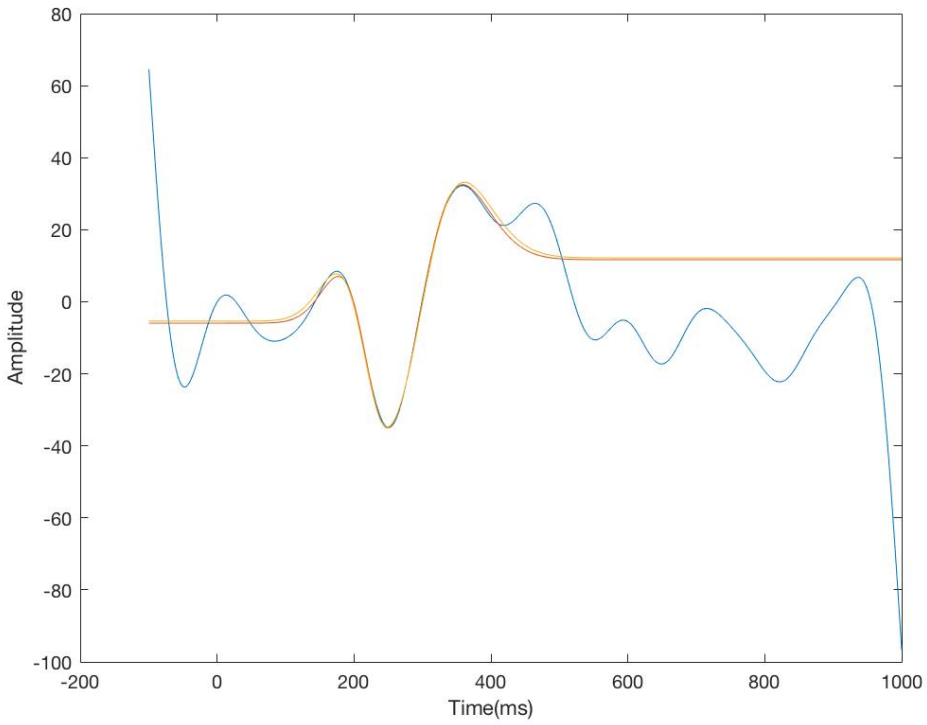

Figure 17: Curve obtained by taking the estimated mean considering a Gaussian distribution

The choice of approximating the histograms with Gaussian distributions is totally arbitrary and was made just to illustrate the results by plotting a curve. In the further study, the posteriors obtained can be studied as they are or by trying to fit a distribution on it (whether it be Gaussian, log-normal...) depending on the interest in doing so. Among the six parameters, I believe that the fourth, i.e. the offset parameter should not be a very significant point to study as it just compensates the electrical background. All other parameters can have a link with the cognitive model that will be associated and linked to this study, but there was no time left to get my hands on this part of the project.

Conclusion

All this study conducted during three months represents the first steps of a much wider project that have ambitions to create a link between a neuro-electric model and a cognitive one. It is difficult to evaluate the quality of the model as long as the link with the cognitive model is not done, but by looking at the curve fitting, the model seems quite accurate. There are many improvements that still remain to do, especially on the MCMC algorithm by changing parameters or by considering the auto-correlation in the signal (the implementation of a Gaussian process could be a lead).
The same methodology can be applied for the study of P300 and Readiness Potential and the same algorithms should be efficient for these two ERPs as well.

It was a real pleasure to work in the CIDLAB with my advisor Joachim Vandekerckhove and my research colleague Thanassi Bakis. Even if I did not have the opportunity to study the cognitive model, the creation of an entire model for EEG data has been a great opportunity to discover the research approach on a fascinating subject. I had the opportunity to be free in my research, hypotheses and choices while I was guided and directed in the good direction. I learned a lot during this research opportunity and being able to apply all the previous years studying mathematics on a very practical subject has been a very interesting discovery and makes me wonder about the possibility of further study in the psychology field.

References

[1] Wei Wu, Chaohua Wu, Shangkai Gao, Baolin Liu, Yuanqing Li, and Xiaorong Gao. Bayesian estimation of erp components from multicondition and multichannel eeg. *NeuroImage*, 88:319–339, 2014.

[2] E Niedermeyer and F.L da Silva. *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*. Lippincott Williams Wilkins, 2004.

[3] Joachim Vandekerckhove. A cognitive latent variable model for the simultaneous analysis of behavioral and personality data. *Journal of Mathematical Psychology*, 60:58–71, 2014.

[4] Salil H Patel and Pierre N Azzam. Characterization of n200 and p300: selected studies of the event-related potential. *International journal of medical sciences*, 2(4):147, 2005.

[5] Hans H. Kornhuber and Lüder Deecke. Hirnpotentialänderungen bei willkürbewegungen und passiven bewegungen des menschen: Bereitschaftspotential und reafferente potentiale. *Pflügers Archiv: European Journal of Physiology*, 1965.

[6] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168, 1944.

[7] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[8] Michael Dawson Nunez. *Refining understanding of human decision making by testing integrated neurocognitive models of EEG, choice and reaction time*. PhD thesis, UC Irvine, 2017.

[9] Dani Gamerman and Hedibert F Lopes. *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. Chapman and Hall/CRC, 2006.