

Electroencephalogram Source Localization based on Dipole Modelling and Particle Swarm Optimization

LAURA PRADA GRACIA

Department of Signals and Systems
Division of Biomedical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2007
Report No. EX088/2007

Abstract

The electroencephalography technique is a valuable tool for diagnosis in the environment of clinical medicine. Through advanced technology, clinical applications systems have been developed for integrating EEG. These systems use dipole modelling amongst other techniques. Dipole modelling or dipole source localization consists of estimating the source localizations associated with the brain electrical activity. In epilepsy, this information can be very useful for clinical applications, where accurate information about the location of an epileptic focus in the brain can be used to plan surgery for its removal.

The goal of this thesis is to develop a method and software for helping in the planning of neurosurgery of epileptic patients. Since determining the source localization through the electric field is an inverse problem, the solution is obtained using a global optimization method called Particle Swarm Optimization (PSO).

The results showed that using a spherical model as a head and an equivalent current dipole as brain activity, general information about source localization can be acquired. Furthermore, using EEG synthetic data, PSO effectively finds the dipole localization with millimetres error magnitude. Finally, the software proposed was tested with real EEG data but the localization accuracy using real EEG signals with epileptic patients couldn't be proved.

Keywords: electroencephalogram (EEG), brain, epilepsy, source localization, equivalent current dipole, EEG forward problem, EEG inverse problem, global optimization method, Particle Swarm Optimization.

Aknowledgements

First of all I would like to thank Prof. Mikael Persson for giving me the opportunity to do this work. It was a pleasure to work in this thesis through which I have discovered the amazing world of the Bioengineering. I would also like to thank Malin C.B. Aberg and Anders Hedstrom because in some sense they have helped me in this thesis development.

Undoubtedly this project is a closure stage, the end of a way where I was guided by many people...

...My friends from the University of Zaragoza. Thank you!!...because during these hard five years YOU were always there, making everything easier.

...My friend from the school, Elena, Leticia, Maria and Alex...because since we met, many years ago, we have shared lot of moments and now you are part of me.

...My Erasmus friends, especially Laura, Natalia and Miguel, the sufferers of this thesis, my psychoanalysts and advisers. Thank you for listening to me.

...My big brother. Thank you for doing the Physic doctorate in Zaragoza and still living at home. Living beside you meant MORE and BETTER.

...My parents, the most important, which have given me this education both academic and moral. Thank you for giving me so much love

Contents

Abstract	I
Acknowledgements	III
Contents	V
Figures Index	IX
Tables Index	XIII
1 Introduction	1
1.1 Motivation	1
1.2 Aim of the thesis and Objectives Outline	2
1.3 Development of the project.	2
1.4 Report Structure	3
2 EEG Background	5
2.1 Source of EEG: Electrophysiological Basis	5
2.2 The electroencephalogram response	6
2.3 Clinical Applications: Epilepsy.	7
2.4 Current methods for EEG analysis and interpretation	7
3 Model Design	9
3.1 The state of the art	9
3.2 Head Model.	10
3.3 Electrode System	11

3.3.1	Introduction	11
3.3.2	The 10-20 Electrode System	13
3.3.3	My Electrode Model.	14
3.4	Source Model	16
4	Forward Problem	17
4.1	Potential Response	17
4.2	Comparison: Model Designed vs DipoleSimulator (BESA)	19
5	Inverse Problem	21
5.1	Introduction	21
5.2	Global Optimization Method	22
6	“Particle Swarm Optimization”	23
6.1	Introduction	24
6.2	Social Behaviour: “Bees’ Swarm”	24
6.3	Development of the PSO.	25
6.3.1	Terminology.	25
6.3.2	Algorithm.	26
6.4	Parameters Values	28
6.5	Space Conditions	29
6.6	¿Why do we use PSO?	30
7	Problem Formulation	31
7.1	Goals and Scope	31
7.2	System Overview	32
8	Tryouts and Results	37
8.1	Introduction.	37
8.2	Particle Swarm Trajectories	38

8.3	Swarm Size	42
8.4	Cognitive/Social Rate	44
8.5	SNR (Signal to Noise Ratio)	46
9	Real Case	51
9.1	EEG Acquisition	51
9.2	Brain Concepts	52
9.2.1	Lobes of the Brain	52
9.2.2	Laterality discrimination.	52
9.3	Procedure description	53
9.4	Results	54
9.5	Discussion	55
10	Conclusions	57
10.1	Technical work and conclusions	57
10.3	Personal conclusions.	58
11	Further Research	59
11.1	Future research lines	59
	References	63
	Images References	67
	Appendix A. Forward Problem	69
	Appendix B. Approximation	75
	Appendix C. BESA	79
C.1	MEGIS Software GMBH	79
C.1.1	What does BESA provide?	79
C.1.2	What does EEGFocus provide?	80

C1.3	DipoleSimulator	80
C.2	DipoleSimulator	80
C.2.1	Program Version	80
C.2.2	Introduction	81
C.2.3	The DipoleSimulator Window	81
C.2.4	Settings	82
C.2.5	Heads Frame	84
C.2.6	Dipole Maps	85
C.2.7	Output Data Waveform Display	85
C.2.8	Map Area	86
C.2.9	How noise is defined	87
Appendix D.	Source Code	89
D.1	10-20 Electrode Source Code	89
D.2	Potential Equation Source Code	91
D.3	Particle Swarm Optimization Implementation	96
D.3.1	Code Structure	96
D.3.2	How to run the PSO Code?	97
D.3.3	PSO Structure Sketch	98
D.3.4	PSO Source Code	99

Figures Index

Figure 2.1.	Section of the head	6
Figure 2.2.	Examples of epilepsy EEG waveform	6
Figure 3.1.	A spherical head model containing four regions.	10
Figure 3.2.	Central cross-section of the four-shell spherical model.	10
Figure 3.3.	Waveguard cap system	12
Figure 3.4.	International 10-20 system views	13
Figure 3.5.	Bipolar and Unipolar measurements	14
Figure 3.6.	Cap Model	14
Figure 3.7.	Spherical view	15
Figure 3.8.	Plan view and Frontal view	15
Figure 3.9.	Current Dipole.	16
Figure 4.1.	Electrodes Potential	18
Figure 4.2.	Current Dipole Moment Response.	18
Figure 4.3.	Graphical comparison	20
Figure 6.1.	“Bees Finding Flowers”	25
Figure 6.2.	Particles movement	27
Figure 6.3.	PSO diagram	27
Figure 6.4.	Absorbing Walls	29
Figure 7.1.	Problem Sketch	32
Figure 7.2.	Four concentric-shells head model	32
Figure 7.3.	10-20 Electrode System	33
Figure 7.4.	Coordinate system for dipoles in spherical model head	33
Figure 8.1.	Example of PSO trajectories with 10 particles	39

Figure 8.2	Fitness values versus number of iterations	40
Figure 8.3	Best fitness and Computational time (2,5,7,10 particles)	43
Figure 8.4	Location error dispersion for 2,5,7,10,20 and 30 particles	43
Figure 8.5	Dipole location trajectories: Cognition-Only , Social-Only	45
Figure 8.6	Best fitness returned by three PSO runs with different velocity update models	46
Figure 8.7	Particle trajectories for different SNR	47
Figure 8.8	Best fitness (5 SNR, 10 SNR, 15 SNR, 20 SNR, No noise)	47
Figure 8.9	Location error dispersion (5 SNR, 10 SNR, 15 SNR, 20 SNR, No noise)	48
Figure 8.10	Localization error according to the increase of noise level	48
Figure 9.1.	Brain Lobes	52
Figure 9.2.	Primary Motor Cortex and Primary Somatosensory Cortex	53
Figure 9.3.	EEG signal	53
Figure 9.4.	Right source location	54
Figure 9.5.	Left source location	55
Figure A.1.	Coordinate system for dipoles in spherical head model	67
Figure A.2.	Four concentric-shells head model	69
Figure A.3.	Dipole within the four concentric-shells model	70
Figure B.1.	Values of c_n with respect to n	73
Figure B.2.	Values of $c_n \cdot f^{n-1}$ with respect to n and $f=0.85$	74
Figure C.1.	Dipole Simulator Window	79
Figure C.2.	Head Model Parameters Box	80
Figure C.3.	Sensor Systems	80
Figure C.4.	Reference Box	81
Figure C.5.	Heads Frame	82
Figure C.6.	Source Parameters Box	82
Figure C.7.	Dipole Maps	83
Figure C.8.	Output Waveform display using Standard 10-20 (19 sensors)	83
Figure C.9.	Map Area	84
Figure C.10.	Map Areas. Example of left and front viewpoints	84

Figure D.1.	PSO Structure Sketch	94
-------------	--------------------------------	----



Tables Index

Table 3.1.	Layers thickness and conductivity.	11
Table 4.1.	Root-mean-square error: mean and Standard deviation.	19
Table 6.1.	Suggested parameters.	29
Table 7.1.	PSO parameters.	34
Table 7.2.	Dipole location search space.	35
Table 7.3.	Current dipole moment search space.	35
Table 8.1.	Inverse problem goal.	38
Table 8.2.	Comparison between the localized dipoles results and the exact values	40
Table 8.3.	Root Mean Square Errors: Location and Moment.	41

Chapter 1

Introduction

This chapter presents the main information related to this thesis, explaining the motivation of this work, the different stages in the project and the goals set. Finally, a description of the rest of the document is introduced.

1.1 Motivation

The EEG (electroencephalography) technique is a valuable tool for diagnosis in the environment of clinical medicine, in particular in neurology, neurosurgery and in psychiatry. EEG visual analysis requires a relatively long training (at least 6 to 12 months) to obtain moderate levels of performance and also appears to be a time consuming process, roughly 10 minutes per EEG. Developing a new technology, that could improve on analyzing and interpreting the EEG information definitely has a considerable cost saving effect.

Through advanced technology, clinical applications systems for EEG have been developed. These systems use source reconstruction, dipole modelling and density reconstructions amongst other techniques, for integrating EEG data. Improved EEG analysis software will enable a faster and more precise diagnosis of the non-invasive EEG in the near future.

Dipole modelling or dipole source localization, consists of estimating the source localizations associated with the brain electrical activity. In epilepsy, this information can be very useful for both clinical and research applications. *Seizures* (a sudden, excessive discharge of nervous-system electrical activity that usually causes a change in behaviour) in up to 30% of people with epilepsy do not respond to available medications. Therefore, accurate information about the location of an epileptic focus can be used to plan surgery for its removal.

1.2 Aim of the thesis and Objectives Outline

The method described in this thesis is a quantitative analysis for estimating the source localization. It consists of transforming the scalp EEG into a source localization related with the generating brain region, representing them by dipoles.

In the first step, in order to estimate the electrical source location in the brain it is necessary to assume a source model and a head model. The head is considered as an sphere which contains four concentric layers with different conductivity values representing the skull, scalp, cerebrospinal fluid and brain. The active neurons are modelled as current dipoles. The second step consists of choosing an appropriate electrode system to work with.

Once a source model, head model and electrode system have been assumed, the next step is to determine the EEG forward problem. It consists of calculating the scalp potential that results from the current source within the head. BESA software, a program for source analysis and dipole localization in EEG and MEG research, will be used in order to compare these results.

The next step is strictly the goal of this thesis. It is to calculate the inverse solution for the source localization. The inverse solution or inverse problem consists of determining the source localization through the electric field (EEG signal). Since the calculation of an inverse dipole solution is a nonlinear problem a global optimization method called PSO (Particle Swarm Optimization) is proposed. PSO, supported on Swarm intelligence, is a technique based on collective behaviour. These systems are constituted by a population of particles which interacting to each other usually they converge to a global behaviour.

Finally, after translating all this theoretical concepts into a program language the results and later discussions will be offered.

1.3 Development of the project

This project consisted of two clearly differentiated parts: the theoretical part and the implementation part.

The theoretical part dealt with acquiring the state of the art in the biomedical concepts and the PSO method. This stage took long time, approximately half of the thesis development period. Once getting all these concepts, the second part consisted of designing and implementing all the acquired knowledge. This period ended with the development of a program and its corresponding test.

1.4 Report Structure

This report is structured in the following way:

Chapter 1: Introduction. The current chapter gives a short introduction about this thesis, the motivation and the goals that will be tried to achieve.

Chapter 2: EEG Background. It introduces the reader to the main concepts about EEG, clinical applications and EEG interpretation.

Chapter 3: Model Design. It introduces the theoretical concepts about the head model, source model and electrode system used in the thesis.

Chapter 4: Forward Problem. In this chapter the EEG forward problem is solved and compared with a clinical application.

Chapter 5: Inverse Problem. In this chapter the inverse problem and the need to use a global optimization method is described.

Chapter 6: Particle Swarm Optimization. This chapter presents the global optimization method used in this thesis.

Chapter 7: Problem Formulation. It presents an overview of the optimization problem dealing with and describes the main parts of the designed solution.

Chapter 8: Tryouts and Results. In this chapter several tryouts are proposed in order to test the implemented program. The results are presented and analysed.

Chapter 9: Real Case. In this chapter the proposed algorithm is used with real EEG measurement data.

Chapter 10: Conclusions. It summarizes the work done during this thesis and exposes the main conclusions extracted both in the technical side and in the personal one.

Chapter 11: Further Research. In this chapter future lines of research are suggested in order to extend and improve the work done in this thesis.

Furthermore, several appendices have been added in order to complement the information provided in this thesis.

Appendix A: Forward Problem. It presents how to solve the forward problem.

Appendix B: Approximation. In this appendix the approximation process to solve the forward problem has been explained.

Appendix C: BESA. This appendix introduces the reader to *BESA software (Brain Electrical Source Analysis)*.

Appendix D: Source Codes. It contains information about the *C* and *Matlab* codes used in this thesis.

Chapter 2

EEG Background

This section provides an introduction to the EEG technique. The first recording of the electric field of the human brain was made by the German psychiatrist Hans Berger in 1924 in Jena. He gave this recording the name *electroencephalogram (EEG)*.

2.1 Source of EEG: Electrophysiological Basis

The EEG is a recording of brain activity. This record is the result of the activity of thousands of neurons in the brain. The exact origin of the EEG is still not completely understood, but it is generally assumed that the measured responses are generated by neurons in the cortex. The electrical activity of the brain can be explained by changes in membrane polarisation, which in turn cause the action potentials.

The electrical potential is conducted through brain tissue, enter the membranes surrounding the brain, CSF (cerebrospinal fluid) and continue through the skull to appear finally at the scalp, as it is showed on figure 2.1.

Some of the current takes the shortest route between the source and the skin by travelling within the dendritic trunk. This intracellular current is called *primary current*. Conservation of electric charges imposes that a current loop flows even through the most distant part of the volume conductor. This extracellular current is known as *secondary or return current*. Both primary and secondary currents contribute to electric scalp potentials.

Although cortical macrocolumns of neurons are assumed to be the main contributors to EEG signals [1], some authors have reported scalp recording of deeper cortical structures including the hippocampus [2], cerebellum[3], and thalamus [4][5].

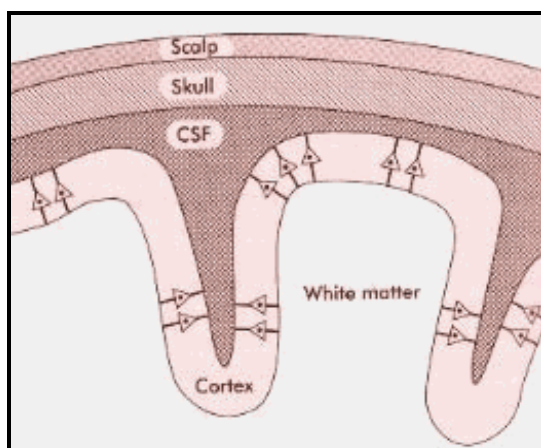


Figure 2.1. Section of the head

2.2 The electroencephalogram response

An EEG is recorded by positioning some electrodes on the scalp. Generally several EEG channels, each one corresponding with an electrode position on the scalp, can be displayed simultaneously. The results of the EEG signals, are recorded in high-resolution real time, amplified and next displayed on paper or on a monitor. The resulting set of signals is highly complex, non-stationary and extremely noisy.

The pattern of the brain activity changes with the level of a person consciousness and brain activity. Typical EEG values, measured on the scalp are in the 20-100 μV range and about 1-2 mV range in case of measuring on brain surface. The bandwidth of this signal is from under 1 Hz to around 50 Hz.

Evoked potentials are those components of the EEG that begin due to a stimulus (which may be electric, auditory, visual, etc). These signals are usually below the noise level and therefore it is not easily distinguished. Larger values are recorded in epilepsy and other disorders (figure 2.2).

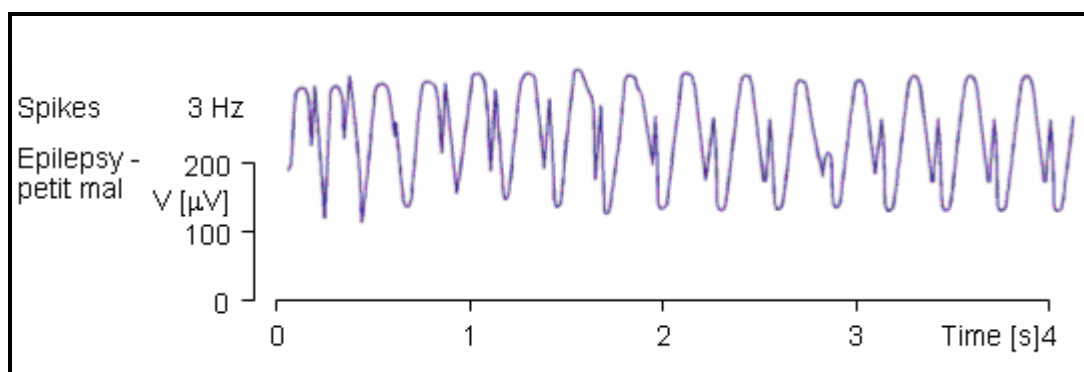


Figure 2.2. Examples of epilepsy EEG waveform.

2.3 Clinical Applications: Epilepsy

Although the origin of EEG responses is not completely brought to light, the signal itself proved to be a valuable tool for diagnosis in the environment of clinical medicine, in particular in neurology, neurosurgery and in psychiatry

EEG is a process non-invasive. In medicine this characteristic involves two meanings. First of all, it is a medical procedure which does not penetrate or break the skin or a body cavity. On the other hand, an abnormal tissue growth, like a neoplasm or tumour, doesn't spread to the surrounding healthy tissue. Since these properties, EEG is used for many purposes like sleep research or diagnosis of brain disorders, such as epilepsy, seizure disorders, tumours, brain trauma and haematomas, among others.

Epilepsy is one of the most common neurological diseases: about 0.5 to 1% of the population suffers from it. It is likely that around 60 million people in the world have epilepsy at any one time. EEG has had tremendous success studying epilepsy. It is able to detect abnormalities in waveforms, such as spikes, sharp waves and spikewave discharges. However, EEG recordings still require additional investigations in studying epilepsy.

In most cases the EEG is considered to be a sensitive rather than a specific diagnostic instrument, making it a suitable instrument to monitoring the course of a disorder on the one hand and to determining a prognosis of the abnormality on the other. In general, one should not try to derive *etiologic diagnoses* from the EEG.

2.4 Current methods for EEG analysis and interpretation

Interpreting EEG records is difficult by an incomplete clinical knowledge. Visually analyzing the raw data and quantitatively examining the time series are the two methods that are available today.

In the first method, *visually analysis*, EEG record is examined visually by the clinical neurophysiologist. The analysis is performed systematically; data are analyzed using pattern analysis methods to associate characteristic differences in the data with differences in patient populations or experimental paradigms. This technique is an empirical science and requires a considerable amount of clinical, in particular neurological, knowledge. Therefore trained physicians have to be involved for the indispensable interpretations, while trained EEG technicians perform the description of the recordings.

As it was said at the beginning of this thesis, visual analysis requires a relatively long training (at least 6 to 12 months) to obtain reasonable levels of performance and also appears to be a time consuming process, roughly 10 minutes per EEG. Developing a new technology that could improve on interpreting the EEG information definitely has a significant cost saving effect.

On the other hand, whole head mapping and source montages enhance the diagnostic potentials of the non-invasive EEG considerably. This is the field of a *quantitative analysis*. *Neuroscan software* [17] and *MEGIS software* [16] are based on this method. Through advanced technology, there are companies which develop products (software and hardware) dedicated to understand the human brain and nervous system. Some specific products developed are focused on clinical systems for EEG and EMG applications. These products use source reconstruction tools which include dipole modelling, dipole scans, and current density reconstruction for integrating EEG and MEG data.

Quantitative analysis can help in clinical analysis, but so far has proved unable to replace visual assessment.

Chapter 3

Model Design

This chapter describes the design procedure. The first part is an overview about typical models used by researches. The following parts describe the head model, the electrode system and the source model employed in this thesis.

3.1 The state of the art

Over the past several years authors have been researched the locations of the cortical sources through developing and getting better new head models. The sources have been modelled as current dipoles and the head has been modelled either as a homogeneous sphere or as a concentric shell structure. Initially, this concentric shell model consists of a homogeneous sphere of neural tissue surrounded by two concentric spherical shells of differing electrical conductivities representing the skull and scalp. These models have been refined further by changing the skull and scalp thickness, increasing the number of layers or adding inhomogeneous regions in the brain (figure 3.1.a).

More realistic head models can be created using finite elements or boundary elements [6]; an example is shown in figure 3.1.b. This finite element or boundary-element model can be closely adjusted to a real head however this model requires more effort [6].

The head model should be selected with caution due to it requires a trade-off between localization accuracy needed and the effort required to developing and working with it.

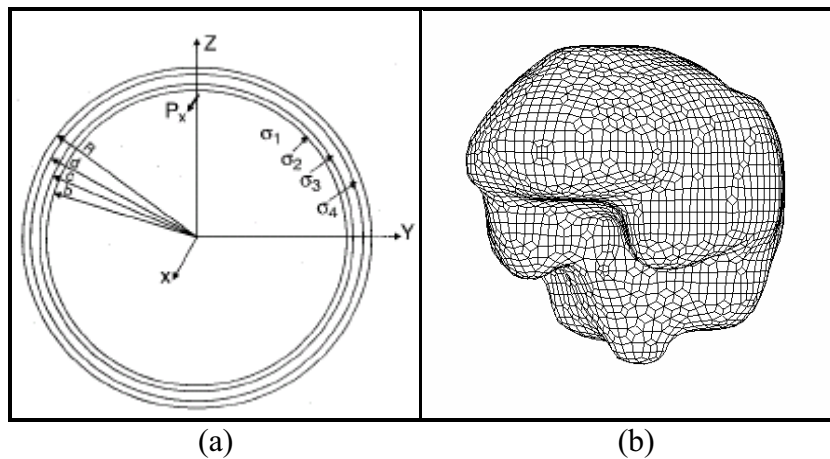


Figure 3.1. (a) A spherical head model containing four regions. The radii and the conductivities (σ) of the model can be set to represent various tissues in the head. A dipole pointing in the $+x$ direction is showed on the z -axis. (b) Boundary element model of the head.

3.2 Head Model

Recent studies showed that EEG dipole source localization can be made as accurate as desired if adequate head models are used. If only general information about a source is needed (i.e., the side of the brain in which it is located) then a simple spherical model can usually be used.

In this project, the model consists of a sphere which contains four concentric layers. Each one with different conductivity values representing the skull, scalp, cerebrospinal fluid (CSF) and brain, as illustrated in figure 3.2.

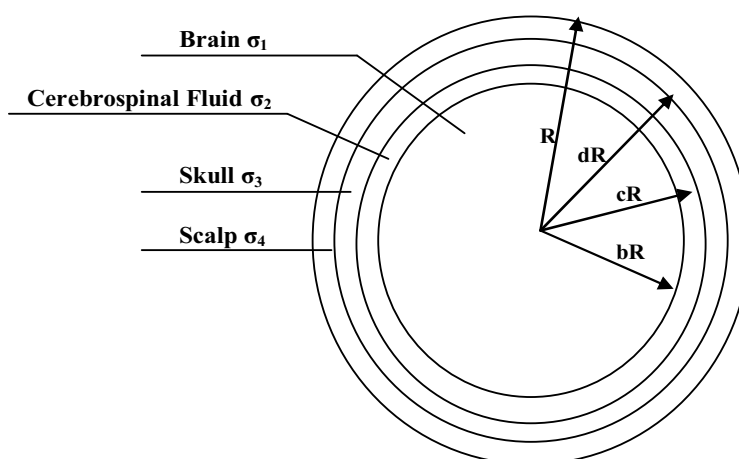


Figure 3.2. Central cross-section of the four-shell spherical volume conductor model. The brain, cerebrospinal fluid, skull and scalp regions are depicted on this figure and their respective conductivity values are labelled by σ_i , $i=1,2,3,4$. The boundaries of these regions are located at bR , cR and dR , where R is the radius of the head model.

Theoretical studies [7] have shown that neglecting the skull in a spherical model of the head can produce localization errors as large as 0.7 cm. Therefore, most EEG head models include a skull region.

A difficulty in this development is that the conductivities of the various tissues in the head can not be measured in the living patient or subject. Fortunately, a theoretical study [8] has found that variations in the following assumed tissues conductivities, which are within physiologically reasonable ranges, will produce localization changes with a maximum difference of approximately 0.4 cm.

Another issue to take into account is the layers' thickness. Studies of local variations in skull and scalp thickness [9] have shown that these variations cause localization errors that are generally much less than 1cm.

Table 3.1 lists layers thickness and conductivity [10][11][12].

	<i>Brain</i>	<i>CSF</i>	<i>Skull</i>	<i>Scalp</i>
<i>Radius [cm]</i>	8.5			
<i>Thickness [cm]</i>		0.2	0.7	0.6
<i>Conductivity [mho/m]</i>	0.33	1.0	0.0042	0.33
<i>Total Sphere Radius = R =10 cm</i>				

Table 3.1. Layers thickness and conductivity

3.3 Electrode System

This section describes the used electrode placement system. First of all an overview about electrode systems will be introduced. Then, the 10-20 system will be reviewed and after that the sensor model developed in this thesis will be explained.

3.3.1 Introduction

In 1947, at the first International EEG congress, it was recognised the need of a standard method of electrodes placement used in electroencephalography. H.H. Jasper, in 1958, defined the 10-20 electrode system, which has become the de-facto standard for clinical EEG. Due to the advancement of multi-channel EEG hardware system, a new standardisation of a larger number of channels was developed. In 1985, the original 10-20 system was modified with an increase of the number of electrodes from 21 to 74 [13]. This new system, called the "10% system" or "10-10 system", has been accepted and is currently used as the standard of the International Federation of Societies for Electroencephalography and Clinical Neurophysiology.

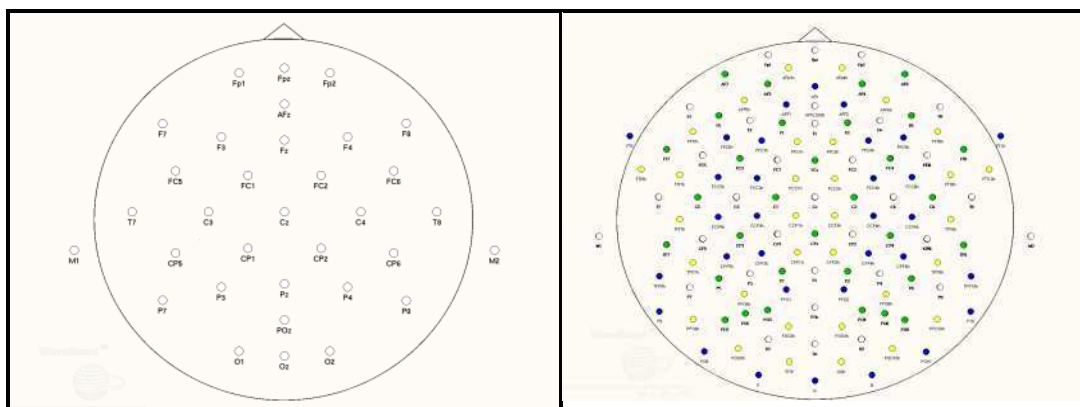
Researchers are moving to a higher number of channels [14], 128 and 256 channels are commercially available nowadays. For this number of sensors there isn't a standard, therefore, there is a recommendation to identify this new systems with a number of landmark electrodes corresponding to the standard site within the 10-20 system.

Obviously more sensors mean more data and information acquired. On the other hand, the closer the electrodes are to each other, the smaller the part of the sensitivity that locates within the brain region. Locating the electrodes closer to each other causes the lead field current to flow more within the skin region, decreasing the sensitivity to the brain region and increasing the noise [15].

In this thesis the 10-20 sensor system has been used because of two main reasons. The first one is that it's a standardised electrode location system and the latter one is that this electrode system is easily to expand.



(a)



(b)

(c)

Figure 3.3. (a) Waveguard cap system. (b) 32 electrodes, white coloured on cap. (c) 128 electrodes, white, green, yellow and blue coloured on cap.

3.3.2 The 10-20 electrode system

The *10-20 System of Electrode Placement* is a method used to describe the location of scalp electrodes. These scalp electrodes are used to record the EEG using an electroencephalograph.

In this system 21 electrodes (19 voltage electrodes and 2 reference electrodes, A_1 and A_2) are located on the surface of the scalp, as shown in figure 3.4.a and 3.4.b. The positions are determined by four reference points: reference points are *nasion*, which is the delve at the top of the nose, level with the eyes; *inion*, which is the bony lump at the base of the skull on the midline at the back of the head; and two preauricular points. From these points, the skull perimeters are measured in the transverse and median planes. Electrode locations are determined by dividing these perimeters into 10% and 20% intervals. Three other electrodes are placed on each side equidistant from the neighbouring points, as shown in figure 3.4.b.

The nomenclature for the electrode locations follows a couple of simple rules:

- Electrode names consist of a single or multiple letters, combined with a number.
- Electrodes on the left are numbered *odd*, electrodes on the right are numbered *even*.
- Electrodes on the centre (midline) are ended with the letter *z* or number *0*.
- Electrodes near the midline (the zero-line) have the smallest numbers, and they increase towards the side
- The letter indicates the location on the head: **Fp**: *frontal pole*; **F**: *frontal*; **C**: *central*; **T**: *temporal*; **P**: *parietal*; **O**: *occipital*; **A**: *ear lobe*

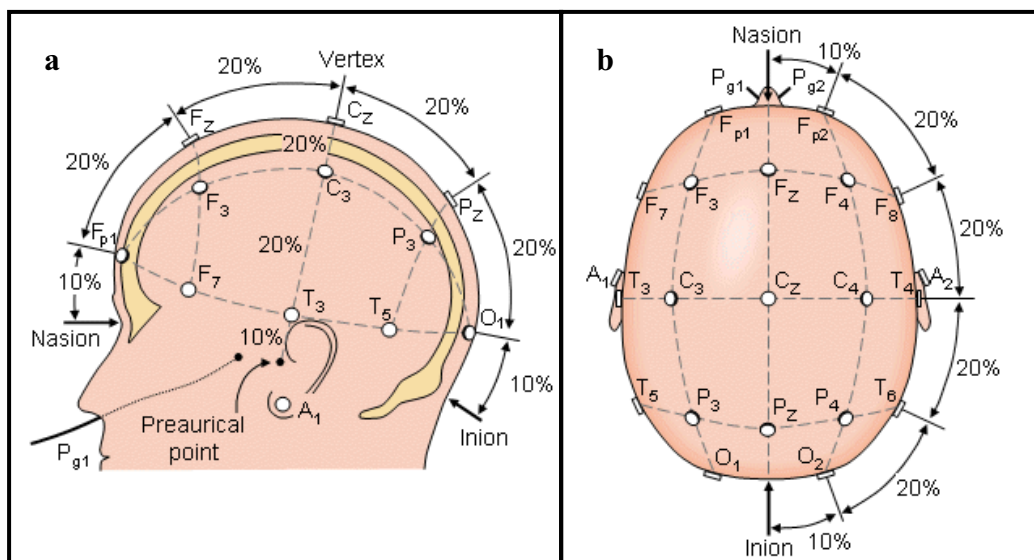


Figure 3.4. International 10-20 system views (a) Left view. (b) Above the head view

Two different kinds of electrodes are used in the EEG measurements, bipolar or unipolar electrodes. In the first method the potential difference between a pair of electrodes is

measured. In the latter method the potential of each electrode is compared either to a neutral electrode or to the average of all electrodes (figure 3.5).

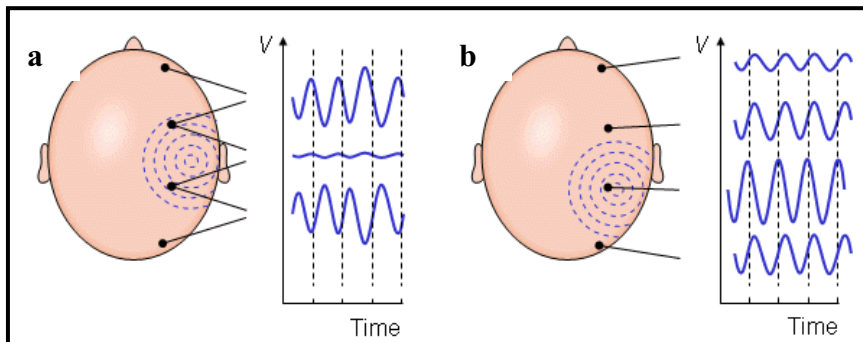


Figure 3.5. (a) Bipolar and (b) Unipolar measurements. Note that the waveform of the EEG depends on the measurement location.

3.3.3 My Electrode Model

In order to calculate the electrodes position, previously we should take into account some measures. The first one is about the skull perimeter. In an ordinary adult the circumference's perimeter formed by the four reference points (nasion, anion and both preauricular points) around the skull is approximately 55-60 cm. The second one is related to the head radius. As it was explained before, the head modelled as a sphere has a radius of 10 cm.

With both parameters, the cap where all the sensors are set can be defined. This model is sketched in figure 3.6, where the cap is delimited by two dimensions: 40 cm as coronal plane and 60 cm as perimeter.

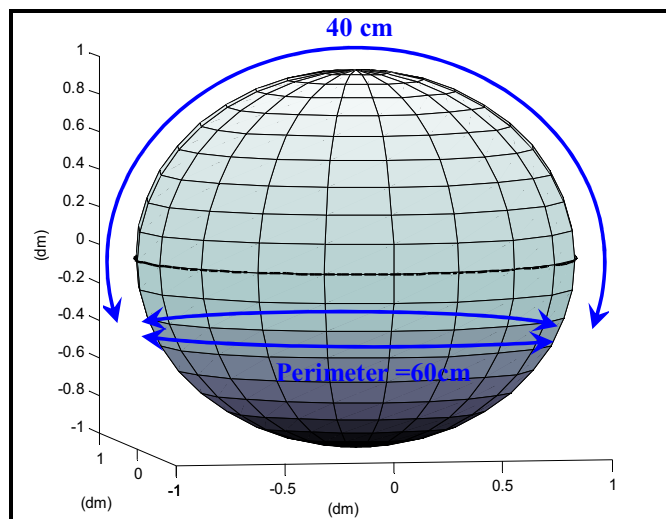


Figure 3.6. Cap Model. (Sphere Radius = 10 cm)

Following the *10-20 System of Electrode Placement*, sensors coordinates were calculated (source code in Appendix D) as figure 3.7 shows. It should be noticed that sensors' placement

is always proportional to the size and shape of the skull. In case of developing another head model such coordinates should be recalculated

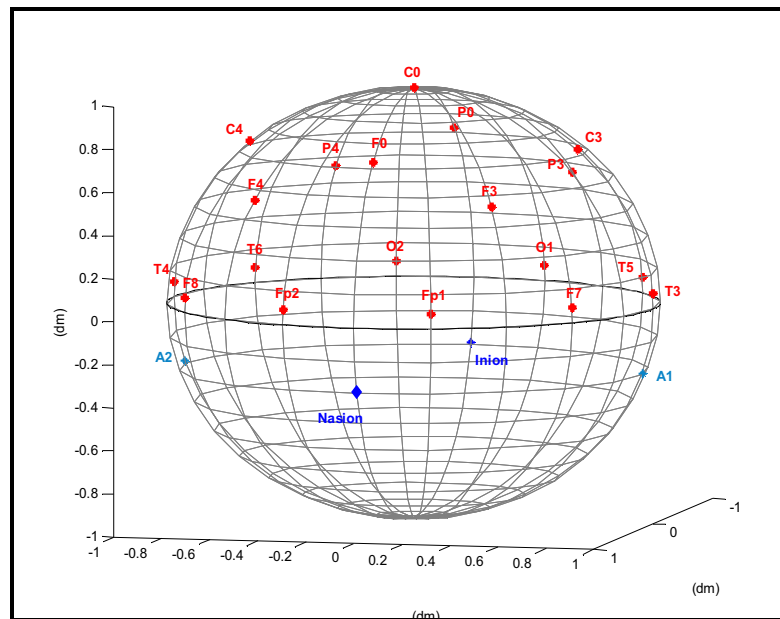


Figure 3.7. Spherical view. The blue dots are the reference points and the red ones are sensors.

The next figures show a plan view and a frontal view.

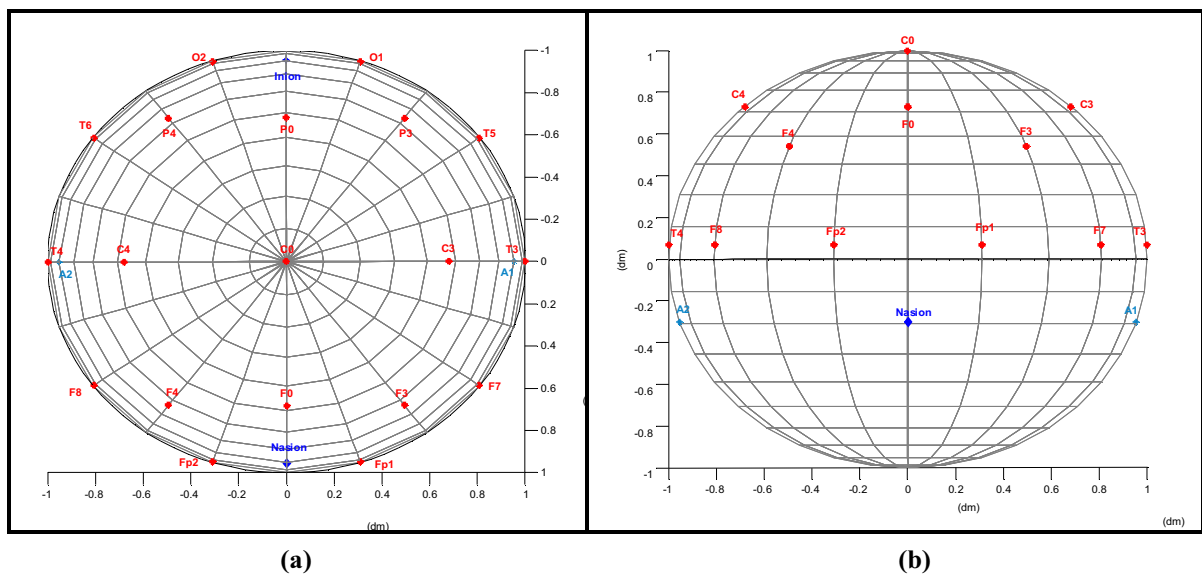


Figure 3.8. (a) Plan view, (b) Frontal view.

The sensors placement is showed clearly in figure 3.8.a:

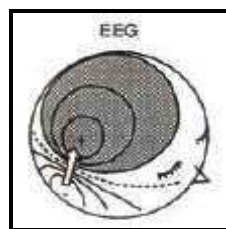
- From left to right along coronal plane: T3, C3, C4, T4
- From the nasion to the inion along sagittal plane: F0, C0, P0

- From right to left along circumferential plane: FP1, F7, T3, T5, O1,O2, T8, T4, F8, Fp2
- Frontal Plane: F4, F3
- Parietal Plane: P4, P3

3.4 Source Model

The electric brain activity can be represented very accurately by an equivalent compound dipole current vector (figure 3.9). This source model has been studied by some authors. The results of simulation studies show clearly that this assumption does not yield large errors [20]. Furthermore, using equivalent dipoles more complicates source systems could be expressed as sums or integrals of these elemental sources.

The magnitude, or strength, of the equivalent dipole is proportional to the number of activated neurons. Therefore it is correlated with the area of activation and the mean dipole current density per square cm. Areas with up to 3 cm in diameter can be very accurately (>99%) modelled by a single equivalent dipole [16].



(a)

$$\text{Current Dipole Moment: } \vec{M} = I \cdot \hat{m} \text{ [A}\cdot\text{m]}$$

(b)

Figure 3.9. Current Dipole. **(a)** EEG field pattern over a spherical head due to a tangential current dipole. The shadowed area indicates positive potential and the white area indicates negative potentials. The dipole is assumed to be in a wall of a cortical fissure. **(b)** Current dipole moment equation.

The dipole current source within the head model will be defined by six parameters: *Dipole location vector*, \mathbf{r} (r_x, r_y, r_z) and *Current dipole moments*, \mathbf{M} (M_x, M_y, M_z). From that moment on let us call *Localization* to the location vector \mathbf{r} and *Orientation* to the unit vector \mathbf{m} .

Chapter 4

Forward Problem

In this chapter the EEG forward problem is solved and compared with a clinical application.

4.1 Potential Response

The EEG forward problem consists of determining over the scalp, the potential that result from current sources within the head. The analytic solution related to a current dipole depends highly on the used head conductor model.

Over the last years, several studies about the EEG forward problem have been developed. Wilson and Bayley derived a closed solution for the potential produced by a current dipole within a homogeneous spherical volume conductor [18]. Rush and Driscoll, using single and homogeneous spheres they presented solutions for both anisotropic and multisphere (three shells) models [19]. Subsequently, two four shell models were presented by Cuffin and Cohen [20] and Stock [21]. More recently, the multishell case of arbitrary spherical shells were provided by Munck [22].

Among these spherical models just the homogeneous model has a closed-form solution. The analytic solutions for remaining models must be represented by infinite sums of weighted Legendre and associated Legendre polynomials [7][23][24][25][19].

In the next equation (1), the analytical solution of the potential using a four layers head model is introduced. For detailed derivation refer to Appendix A.

$$v = \frac{1}{4\pi\sigma_4 R^2} \sum_{n=1}^{\infty} c_n f^{n-1} m \cdot \left[r_0 P_n(\cos \theta) + t_0 \frac{P_n^1(\cos \theta)}{n} \right] \quad (1)$$

The infinite series must be truncated or approximated for the reason that these analytic expressions are computationally expensive. Several approximations have been proposed [7][23][26][27][28]. In this work, to solve the computational problem, approximations based on mathematical analysis of infinite sums of Legendre and associated Legendre polynomials [23] has been developed. In Appendix B, this approximation is given in detail.

Once implemented equation (1), in figure 4.1 the potential field distribution due to a single dipole with a certain response (figure 4.2) is showed. The signals depicted on figure 4.1 are the nineteen sensors of the 10-20 electrode system (source code in Appendix D).

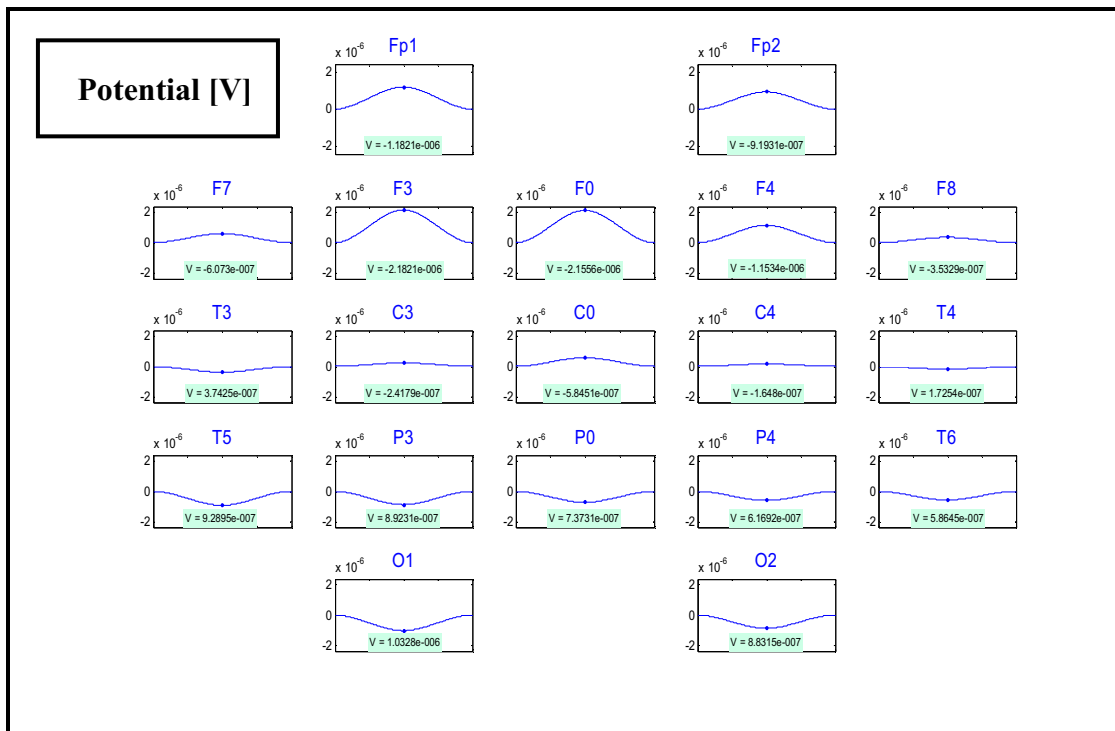


Figure 4.1. Electrodes Potential due to a equivalent dipole within the head model. Equivalent dipole parameters: location $(-0.25, 0.24, 0.42)$, orientation $(0, -0.89, -0.45)$, Intensity (30) .

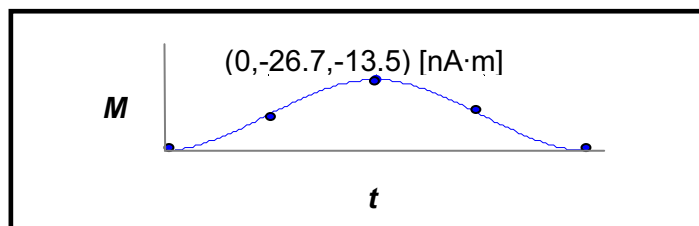


Figure 4.2. Current Dipole Moment Response

A single dipole is used in this example. Nevertheless, the head can be modelled as a linear volume conductor which permits the use of the superposition principle. Therefore, the potential response to multiple sources may be approximated by combining the individual potentials of a group of dipoles at different times, orientations, and locations.

4.2 Comparison: Model Designed vs DipoleSimulator (BESA)

To verify that the developed forward program works properly, it has been compared with a clinical software.

MEGIS Software GmbH is a software which offers advanced technologies for analysis and visualization of human brain activity. This software offers to the users *BESA* (Brain Electrical Source Analysis), a program for source analysis and dipole localization in EEG and MEG research. This program has a tool called *DipoleSimulator* which have been used in this thesis to provide a better understanding of dipole modelling. In order to understand deeply how does BESA work, refer to Appendix C.

The comparison process consists on inserting a dipole with the same localization (\mathbf{r}) and moment (\mathbf{M}) to the program developed (Matlab) and *DipoleSimulator*. The comparison has been reflected by the mean of the root-mean-square error \bar{e}_0 (2), and by the figure 4.3.

$$\bar{e}_0 = \sqrt{\frac{\sum_{m=1}^M (v_m - \bar{v}_m)^2}{\sum_{m=1}^M v_m^2}} \quad (2)$$

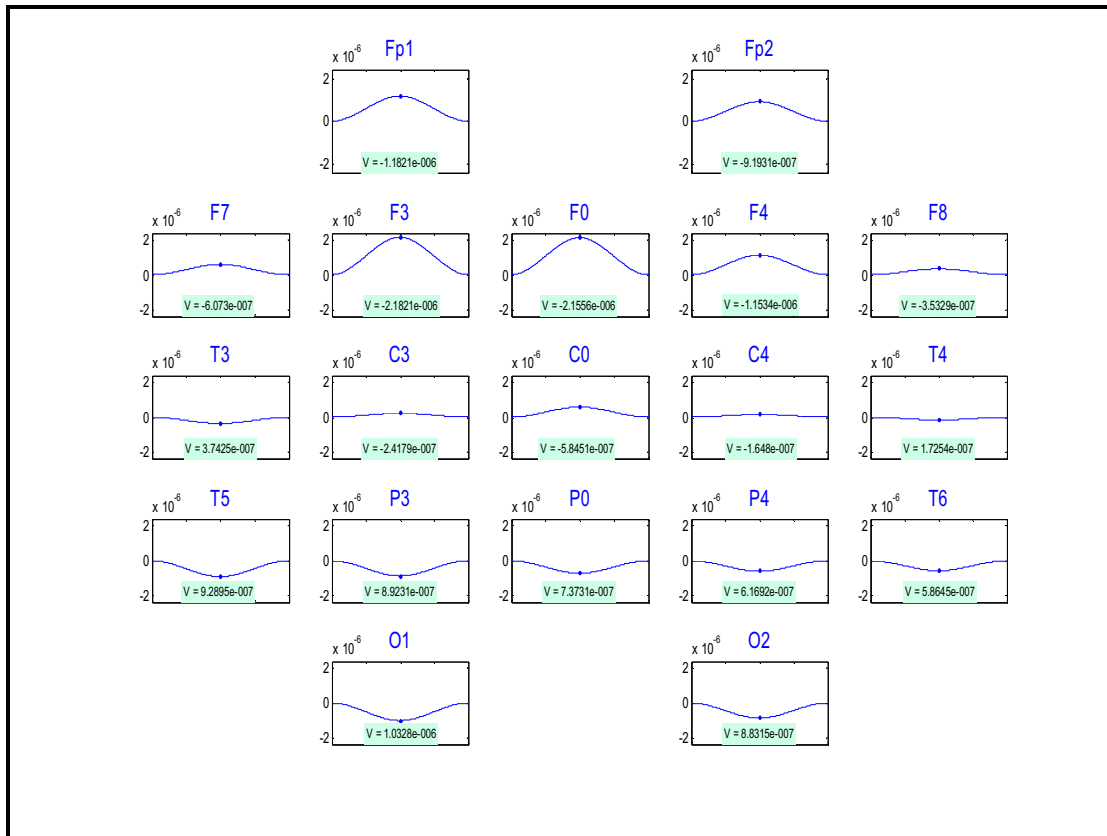
where v is the *DipoleSimulator* potential, \bar{v} is the proposed potential and M is the total number of electrodes.

The root-mean-square error, \bar{e}_0 , has been calculated for several simulations with different dipole localizations. The difference between the *DipoleSimulator* and the code proposed is expressed by the mean and standard deviation of the root mean square error (table 4.1).

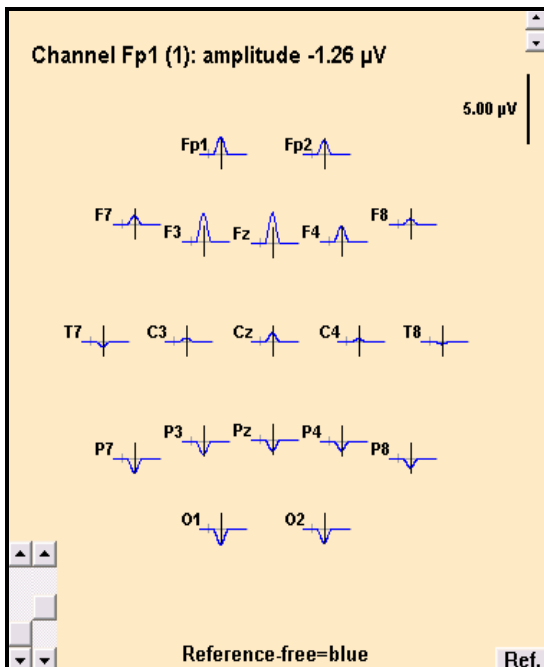
<i>Mean</i>	<i>Standard Deviation</i>
$6.82 \cdot 10^{-2} \text{ V}$	$1.46 \cdot 10^{-2} \text{ V}$

Table 4.1. Root-mean-square error: mean and standard deviation.

As it is showed, the difference between both programs is very small. Therefore, the proposed algorithm which approximates the forward solution of multishell models is precise as a clinical program. Besides, it is computationally efficient taking just a few seconds in computation.



(a)



Potencial [V]			
Fp1	$-1.26 \cdot 10^{-6}$	F3	$-2.07 \cdot 10^{-6}$
F7	$-0.66 \cdot 10^{-6}$	FZ	$-2.19 \cdot 10^{-6}$
T7	$0.36 \cdot 10^{-6}$	F4	$-1.14 \cdot 10^{-6}$
P7	$0.97 \cdot 10^{-6}$	C3	$-0.24 \cdot 10^{-6}$
O1	$1.11 \cdot 10^{-6}$	CZ	$-0.64 \cdot 10^{-6}$
O2	$0.97 \cdot 10^{-6}$	C4	$-0.21 \cdot 10^{-6}$
P8	$0.64 \cdot 10^{-6}$	P3	$0.91 \cdot 10^{-6}$
T8	$0.17 \cdot 10^{-6}$	PZ	$0.75 \cdot 10^{-6}$
F8	$-0.41 \cdot 10^{-6}$	P4	$0.63 \cdot 10^{-6}$
Fp2	$-1 \cdot 10^{-6}$		

(b)

Figure 4.3. Graphical Comparison (a) My Matlab Program (b) Dipole simulator.
Parameters: Location (-0.25,0.24,0.42), Orientation (0,-0.89, -0.45), I (30)

Chapter 5

Inverse Problem

Once the forward problem solved, the next step is to calculate the inverse solution for the source location. In this chapter the inverse problem, its implications and the need to use a global optimization method is described.

5.1 Introduction

The inverse problem consists of determining the source localization through the electric field (EEG). Unfortunately, the signals measured on the scalp surface don't directly indicate the location of the active neurons in the brain due to the ambiguity of the underlying static electromagnetic inverse problem (Helmholtz, 1853). That is, many different source configurations can generate the same distribution of potentials and electric fields on the scalp (for a review see Fender (1987)).

However, if physiologically and physically a priori assumptions are introduced, the inverse problem can be solved and estimations of the sources become possible. The more appropriate these assumptions are the more trustable are the source estimations. These a priori assumptions depend on where and how the signals were generated in the brain. Since this information is not known, it is up to the user to decide if the constraints used in a given inverse solution are physiologically plausible.

The basic a priori assumption is to employ one or a few equivalent current dipoles which can adequately model the surface measurement. For example, epileptic activity is assumed to be generated by a fairly limited number of simultaneously sources.

The reason why new methods are continuously developed is mainly that new knowledge of how signals are generated is continuously incorporated as a priori constraints. Thus, the

source localization problem is not yet solved, since not all information about signal generation is yet now [29].

Since the calculation of an inverse dipole solution is a nonlinear problem, the solution is usually obtained by an iterative numerical process. Parametric and imaging methods are the two general techniques used for EEG sources estimation [30]. In this work, it will be developed the first method using a global optimization method called Particle Swarm Optimization (PSO). In the next chapter this algorithm is explained deeply.

5.2 Global Optimization Method

The calculation of an inverse dipole solution is a nonlinear problem. Nonlinear models are everywhere in many applications, e.g., in advanced engineering design, data analysis, process control, scientific modelling, and others. Their solution often requires a global search approach.

The objective of global optimization is to find the globally best solution of (possibly nonlinear) models, in the (possible or known) presence of multiple local optima. Such as an optimization problem arise a large number of continuous variables and a solution cannot be guaranteed in a finite number of steps, there is a need of designing a reliable algorithm to solving this problem.

Stochastic methods have been developed and used by several researchers to solve the above problem. It can be proved that these methods are able to generate a solution close to the global optimum (probability approaches 1) if the procedure is allowed to continue forever. These methods need a probabilistic global search procedure. There are some special local algorithms which starting from several points distributed over the whole optimization region, they cooperate with some learning strategies (instead of randomly generating) in order to guide the search. They are called *Heuristic Search Methods*.

Heuristic tools have evolved in the last decade. They facilitate solving optimization problems that were previously difficult or impossible to solve. One of these tools is the Particle Swarm Optimization technique.

Chapter 6

Particle Swarm Optimization

In the following sections it is explained the PSO technique. After the introduction, PSO will be described first of all by a natural phenomenon example and then by its algorithm and parameters. Finally a conclusion will be provided.

“... Homo sapiens- literally, “intelligent man” – has adapted to nearly every environment on the face of the earth, below it, and as far above it as we can propel ourselves. We must be doing something right...

... What we do right is related to our sociality...

We will investigate that elusive quality known as intelligence, which is considered first of all a trait of humans and second as something that might be created in a computer, and our conclusion will be that whatever this “intelligence” is, it arises from interactions among individuals.

We humans are the most social of animals: we live together in families, tribes, cities, nations, behaving and thinking according to the rules and norms of our communities, adopting the customs of our fellows, including the facts they believe and the explanations they use to tie those facts together. Even when we are alone, we think about other people, and even when we think about inanimate things, we think using language- the medium of interpersonal communication. ...”

James Kennedy and Russell C. Eberhart

Swarm Intelligence [31].

6.1 Introduction

“Why is social behaviour ubiquitous in the animal kingdom? Because it optimizes. What is a good way to solve engineering optimization problems? Modelling social behaviour” [32].

Particle Swarm Optimization (PSO) was developed in 1995 by Kenedy and Eberhart [32], this new stochastic evolutionary computation technique is based on collective behaviours and it has been found to be extremely effective in solving a wide range of engineering problems.

Particle Swarm Optimization has roots in two main component methodologies. The more obvious is the link to artificial life (A-life) in general and to bird flocking, fish schooling, and swarming theory in particular. However, it is also related to evolutionary computation and has links to both genetic algorithms and evolutionary programming.

The optimization potential of simple behaviours has been most noted in studies of insects, and in particular in the behaviours of the social insects. An insect may have only a few hundred brain cells, but insect organizations are capable of architectural marvels, elaborate communication systems and terrific resistance to the threats of nature. Particle Swarm Optimization system is constituted by a population of particles which interacting to each other usually they converge to a global behaviour.

The classic example of a swarm is a swarm of bees, but the metaphor of a swarm can be extended to other systems with a similar architecture. An ant colony can be thought of as a swarm whose individual agents are ants, a flock of birds is a swarm whose agents are birds, traffic is a swarm of cars, an immune system is a swarm of cells and molecules, etc [31].

Although the notion of a swarm suggests aspects in all types of collective behaviour, this thesis is focused on collective motion in space. The swarm will be composed by current dipoles which will move freely within the brain.

6.2 Social Behaviour: Bees' Swarm

A clearly example of collective behaviour is in “Bees Finding Flowers” [33]. Let us describe their conducts. Imagine a swarm of bees flying over a field. Without any knowledge about the field, their objective is to find the location with the highest density of flowers.

They begin in random locations with random velocities. The movement of the bee depends on the locations that it found the most flowers and the location of other bees where they have found an abundance of flowers. Somehow they communicate to each other and they decide if return to the location where it had personally found the most flowers, or exploring the new location found by others. Undecided bees change their directions somewhere between the two points depend on social influence (figure 6.1.a).

Occasionally, one bee may fly over a place with the most flowers. In this case it is probably that the remaining bees change their directions toward that location. Eventually, the whole swarm will fly over the place with the highest flower concentration (figure 6.1.b).

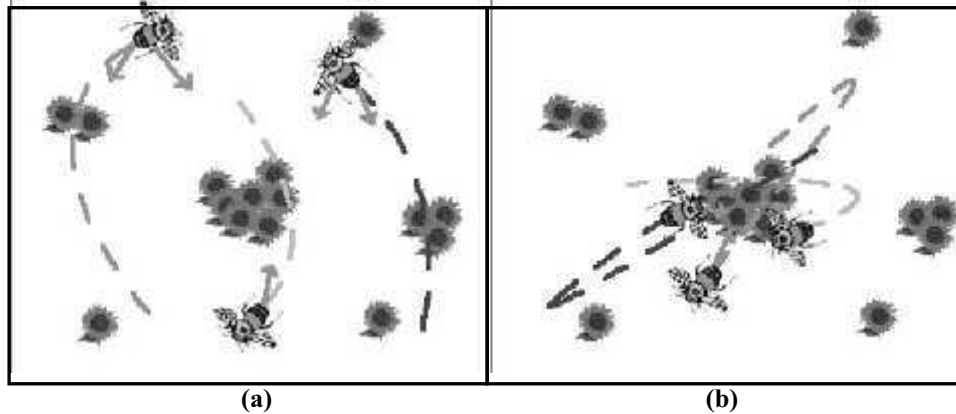


Figure 6.1. “Bees Finding Flowers” (a) Each bee is attracted to the area of highest concentration found by the entire swarm and the best location that itself encountered so far. (b) Eventually, all bees swarm around the best location.

6.3 Development of the PSO algorithm

6.3.1 Terminology

In order to understand properly the PSO algorithm, it is listed below some key words:

Particles or Agents represent each entity in the collective group (swarm) and a probable candidate solution to the optimization problem. Each one has the same goal: try to find the optimized location.

Position is represented by N -dimensional coordinates (bee place in the field, 2-dimension). These set of coordinates correspond to the solution parameters for the problem being optimized.

Fitness is a value which characterizes the “goodness” of a position. In the analogy above *fitness* referred to the density of flowers: the higher the density, the better the location.

pbest or personal best is the location with the highest fitness value personally found by a particle.

gbest or global best is the location with the highest fitness value discovered by the whole swarm.

6.3.2 Algorithm

The main tasks of a PSO algorithm are outlined in five goals [33]:

0. Each particle i maintain a current position, \vec{x}_i , current velocity, \vec{v}_i , and personal best position, $p_{best\ i}$. These vectors, \vec{x}_i and \vec{v}_i , are formed by N components for each dimension in the N -dimensional optimization.
1. **Initialization:** System is initialized with a population of random potential solutions. Each potential solution is assigned a randomized velocity, \vec{v}_i .
2. **Fitness Evaluation:** The fitness function is used for evaluating the current position, $\vec{x}_i(t)$, and returning a fitness value, $F(\vec{x}_i(t))$. In section 6.7, this cost function will be explained in detail.
3. **p_{best} Update:** Each particle keeps track of the coordinates for which it has achieved the best solution so far.
4. **g_{best} Update:** Each particle compare its fitness value with the entire swarm. If the position of the best particle is denoted by the vector g_{best} then

$$g_{best}(t) \in \{p_{best\ 0}, p_{best\ 1}, \dots, p_{best\ S}\} \mid f(g_{best}(t)) = \min\{f(p_{best\ 0}(t)), f(p_{best\ 1}(t)), \dots, f(p_{best\ S}(t))\} \quad (3)$$

where S is the total number of particles in the swarm and f is cost function.

5. **Update the Particle Velocity and Position:** The velocity of each particle- i is changed towards its p_{best} and g_{best} fellows (4). As a result, a new position in the solution space is calculated for each one by adding the new velocity value to each component of the particle's position vector (5).

$$\vec{v}_i(t) = w \cdot \vec{v}_i(t-1) + c_1 \text{rand}(\cdot) \cdot (\vec{x}_{p_{best\ i}} - \vec{x}_i(t)) + c_2 \text{rand}(\cdot) \cdot (\vec{x}_{g_{best}} - \vec{x}_i(t)) \quad (4)$$

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \quad (5)$$

Equation (4) is the key factor of the PSO technique. Let us analyse term by term:

- w is the *inertial weight* (a number between 0 and 1) which control the influence of the original course.
- c_1, c_2 are known as the *cognitive* and the *social* rates. The *cognitive* term c_1 is a scaling factor which determines how much the particle is influenced by his best location, $\vec{x}_{p_{best}}$. Usually it is called “simple nostalgia” because the individual tends to return to the place that most satisfied in the past. On the other hand, the *social* term c_2 refers to how much it is influenced by the rest of the swarm, $\vec{x}_{g_{best}}$.

- $rand()$ is a random function which returns a random number between 0.0 and 1.0. This term symbolizes the unpredictable component of the natural swarm behaviour.

The three adaptive coefficients: *inertial weight*, *cognitive term* and *social term* can be summarized with this sentence: “The better I am, the more I follow my own way. The better is my best neighbour, the more I tend to go towards him”.

The next figure illustrates the velocity update in a 2D parameter space.

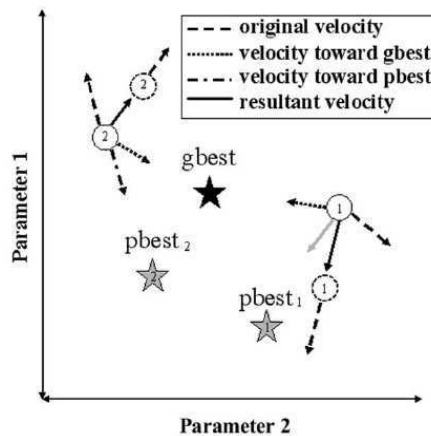


Figure 6.2. Particles movement. Particle 1 and 2 are accelerated toward the location of the global best solution g_{best} , and the location of their own personal best p_{best} , in a 2D parameter space.

The PSO algorithm, described above, is sketched in figure 6.3.

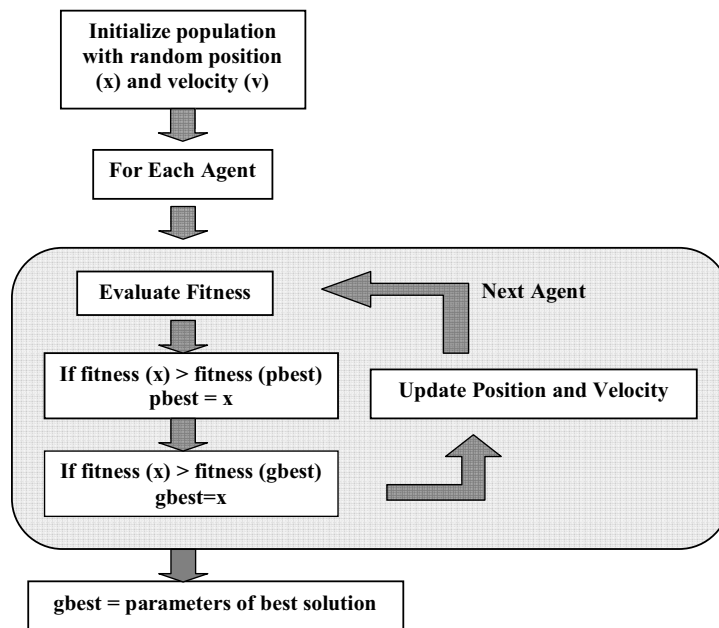


Figure 6.3. PSO diagram

6.4 Parameter Values

Since its first publication, a large number of studies have been done in order to study and improve the PSO technique. These studies concentrated mostly on the PSO control parameters, namely the scaling factors, inertia weight and swarm size [33].

From these empirical studies it can be concluded that the PSO is sensitive to control parameter choices. Wrong initialization may guide to divergent or cyclic behaviour.

In next paragraphs there are suggested some parameter values:

▶ Swarm Size

As we can imagine, there is a trade off between the most exploration of the solution space and the computation time. The higher populations size the most detailed exploration. Fortunately, relatively small population sizes can explore an optimal solution space while avoiding excessive fitness evaluation. Parametric studies have found that a population size of about 30 is optimal [34] and even smaller sizes have been successful for engineering problems.

▶ Inertia weight, w

The inertia weight was introduced by Shi and Eberhart to control the influence of the velocity vector on a particle's position [35][36]. Initial empirical studies of PSO have demonstrated that the value of w is extremely important to ensure convergent behaviour [37]. Eberhart and Shi suggest a range between 0.9 and 0.4 [37].

▶ Acceleration coefficients, c_1 and c_2

Initial PSO studies used $c_1 = c_2 = 2.0$. Although good results have been obtained, it was observed that velocities quickly tended to large values. According to [32] a standard selection of these scaling factors is $c_1 = c_2 = 1.49$.

▶ Asynchronous and Synchronous Updates

There are two different types to PSO algorithm updating, synchronous and asynchronous. In the synchronous update, all the particles move at the same time before the best selection is made. On the other hand, in the asynchronous one the best selection is updating every iteration after each particle movement. The empirical work [38] concludes that asynchronous updates are generally less costly.

The following table summarizes the parameter values which will be used initially in the PSO simulations.

Swarm Size ≤ 30	$w = 0.9$	$c_1 = c_2 = 1.49$	Asynchronous update
----------------------	-----------	--------------------	---------------------

Table 6.1. Suggested parameters

These parameters values should be considered with care, since the corresponding empirical studies are based on only a limited sample of problems.

6.5 Space Conditions

Without any boundaries or limits on the velocity, particles could fly out of the physically solution space. Two constrains are used in order to solve it, V_{max} and *boundary conditions*.

Let us explain briefly these new terms:

▶ V_{max}

For each dimension the velocity is delimited by a maximum velocity, V_{Max} . This factor should be proportional to the dynamic range of each dimension.

$$\text{If } \begin{cases} v_i(t) \geq V_{Max} & \text{then } v_i = V_{Max} \\ v_i(t) \leq -V_{Max} & \text{then } v_i = -V_{Max} \end{cases}$$

▶ Boundary conditions

It is needed a space limitation technique for those particles which leave the search space boundaries. To manage this problem, the authors have developed three different boundary conditions [33]: *Absorbing Walls*, *Reflecting Walls* and *Invisible Walls*.

In the following PSO simulations it will be used the *Absorbing Walls* technique. In this method, when a particle collides with the limit of one of the dimensions, the velocity (in this dimension) is zeroed (figure 6.4).

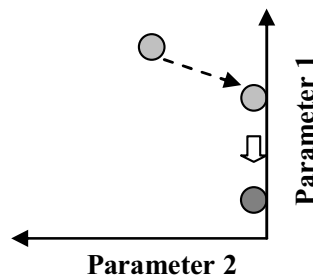


Figure 6.4. Absorbing Walls. The particle reaches the boundary of Parameter 2 and the velocity in this dimension gets zero.

6.6 Why do we use PSO?

In recent studies, particle swarm optimization (PSO) has shown to be an efficient, robust and simple optimization algorithm. Several optimization problems use PSO because of its simple evolutionary principle. Moreover, PSO has technically two key advantages: the velocity updating is simple and its required parameters are easy to tune.

On the other hand, we should take into account two difficulties. The first one is to remind that PSO parameters depend highly on the optimization problem environment. The second one is that the swarm will reach a point of equilibrium under certain conditions [39] [40]. It is a general risk of these kinds of methods; particles can be trapped in undesirable local minima, accepting a certain false location as the correct one.

Chapter 7

Problem Formulation

Up to this point the main theory concepts needed to understand this project have been explained. This chapter provides an overview of the optimization problem that we are dealing with.

7.1 Goals and Scope

Let us begin imagining that we have stored electric field values measured over the scalp of a certain human head (*EEG Signal*). Our problem consists of determining the source location responsible for such a signal (*Inverse Problem*). As we already know, many different source configurations can generate a similar electric field distribution on the scalp. Therefore, some initial suppositions should be introduced (*A Priori Assumptions*). In this project, this basic assumption is to employ just a single active source which will be the only responsible for the surface electric field. In order to reach these goals previously the head was modelled by the *Four Layers Sphere Model* and the sources by *Equivalent Current Dipoles*.

Now we are cable to start moving our single dipole inside the head model, in order to get the best fit (*Cost Function*) between the recorded EEG signal and the estimated values. This process can be trapped in certain false localization and as it was pointed out, an optimization method should be used (*Particle Swarm Optimization*) since we are dealing with a nonlinear problem.

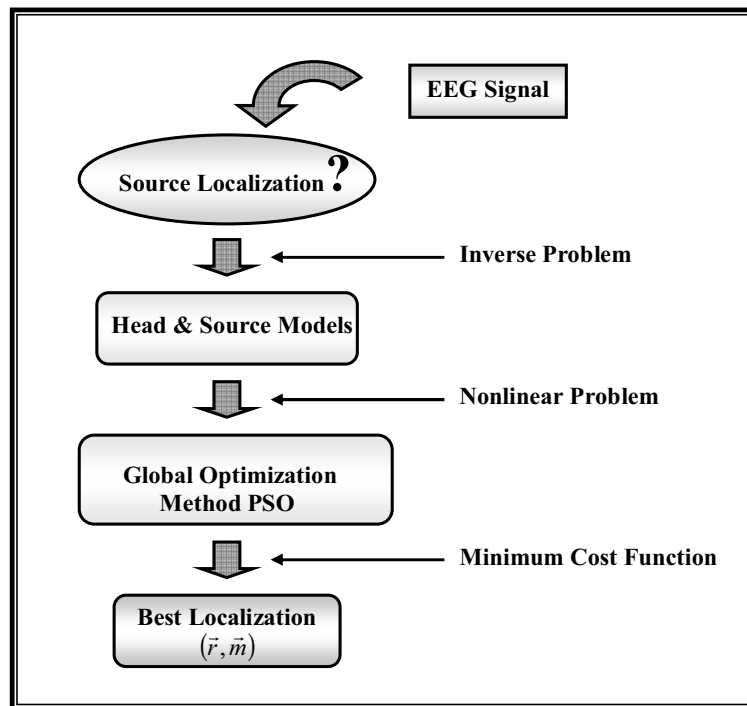


Figure 7.1. Problem Sketch.

7.2 System Overview

A brief outline of the main aspects follows bellow:

► Head Model

A simple spherical model is used. The model consists of a sphere which contains four concentric layers. Each one with different conductivity values representing the skull, scalp, cerebrospinal fluid (CSF) and brain. In figure 7.2 the used conductivity and radius values are showed.

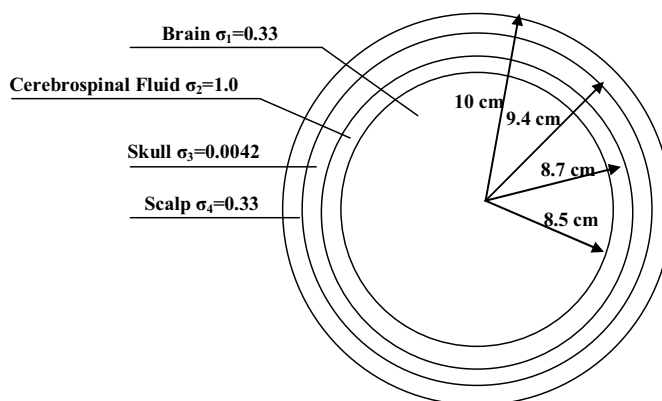
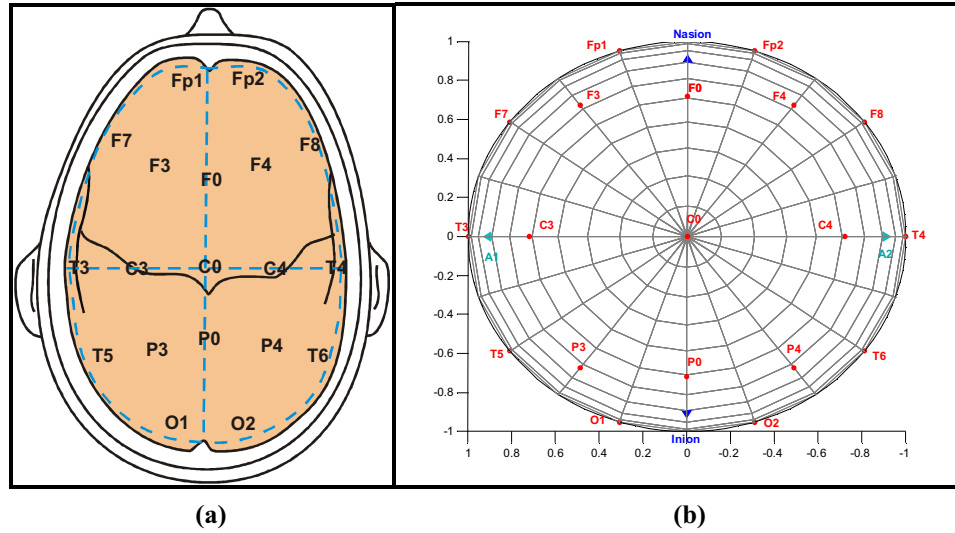


Figure 7.2. Four concentric-shells head model.

► Electrodes System

The standard for clinical EEG *10-20 System of Electrode Placement* has been chosen. It is composed by 19 voltage electrodes and 2 reference electrodes.



► **Global Optimization method**

In order to solve the nonlinear problem (inverse problem), it will be used a new technique called Particle Swarm Optimization. As it was explained it is an efficient and simple optimization algorithm.

The following table summarizes the main concepts:

		<i>Analogy</i>	
		“Bees searching flowers”	“Dipoles moving within a brain model searching the brain activity focus”
Swarm Size	w	c₁ & c₂	Boundary Conditions
≤ 30	0.9	1.49	Absorbing Walls
Velocity Updating			
$\vec{v}_i(t) = w \cdot \vec{v}_i(t-1) + c_1 \text{rand}(\cdot) \cdot (\vec{x}_{pbest\ i} - \vec{x}_i(t)) + c_2 \text{rand}(\cdot) \cdot (\vec{x}_{gbest} - \vec{x}_i(t))$			
Best Solution			
gbest (\vec{r}, \vec{M})			

Table 7.1. PSO parameters.

► **Cost Function**

The objective of the source location searching process will consist of minimizing a cost function. The best solution will be the source localization which achieves the lowest cost function value.

Because of the fact that each current dipole is defined by six parameters (dipole location vector (r_x, r_y, r_z) and current dipole moment vector (M_x, M_y, M_z)), the source searching process becomes a six-dimensional optimization problem.

The cost function F , is defined as

$$F(r_x, r_y, r_z, M_x, M_y, M_z) = \int_0^T \sum_{m=1}^M \left(\left| V_m(r_x, r_y, r_z, M_x, M_y, M_z, t) - V_m^{meas}(\vec{r}, \vec{r}, \vec{r}, \vec{M}, \vec{M}, \vec{M}, t) \right|^2 \right) dt \quad (7)$$

where

- $V_m(\vec{r}, \vec{M}, t)$ is the simulated voltage data from the four layers head model
- $V_m^{meas}(\vec{r}, \vec{M}, t)$ is the measured data from the EEG signals. In chapter 8, in order to explain PSO algorithm, $V_m^{meas}(\vec{r}, \vec{M}, t)$ data are also simulated voltages. Real EEG values will be used in chapter 9.
- M is the total number of electrodes.

► **Search Space**

The six-dimensional search space is defined in the next tables:

<i>Dipole Location</i>
Cartesian Coordinates
Normalized to the sphere radius
(r_x, r_y, r_z)
$r_i \in [-1, 1], i = x, y, z$

Table 7.2. Dipole location search space.

<i>Current Dipole Moment</i>
Cartesian Coordinates
(M_x, M_y, M_z)
$[-40 \text{ nA}\cdot\text{m}, 40 \text{ nA}\cdot\text{m}]$

Table 7.3. Current dipole moment search space.

In the next proposed simulations the intensity range is considered from -40 nAm to 40 nAm. However, rescaling the input parameters (EEG data), any intensity range can be considered because the EEG forward problem is a linear problem.

Chapter 8

Tryouts and Results

In this chapter there are a total of four tryouts. The first one is dedicated to showing how PSO works and the remaining tryouts are focused on understanding the influence on PSO due to different parameter selections.

8.1 Introduction

There are many parameters, associated with the Particle Swarm Optimizer that may affect its performance. There are the social and cognitive learning rates, the population size, synchronous and asynchronous updates and various additional controls, such as inertia and constriction factors. For any given problem, the values of these parameters may have significant impact on the efficiency and reliability of the PSO.

Since the introduction of Particle Swarm Optimization in 1995 by James Kennedy and Russ Eberhart [32], several variations of the basic algorithm have been developed. Each researcher has a favourite implementation (different population size, different inertial weight, and so forth). In the next sections some of these factors have been investigated in the *Source Location problem* with the goal of providing canonical parameters.

In all these tryouts, the real EEG voltage is made with synthetic data. In [45] the authors use the expression “inverse crime” to denote the act of employing the same model to generate synthetic data, as well as to invert. Furthermore, they warn against committing the inverse crime in order to avoid trivial inversion and state that it is crucial that the synthetic data be obtained by a forward solver which has no connection to the inverse solver. In order to avoid this problem a white Gaussian noise is added to the simulated measurement data (tryout SNR).

The proposed tryouts are listed as follows:

- Particle Swarm Trajectories
- Swarm Size
- Cognitive/Social Rate
- SNR (signal to noise ratio)

Goal

Location (r_x, r_y, r_z) Moment (M_x, M_y, M_z)

Table 8.1. Inverse Problem Goal

Since PSO algorithm starts with random localizations, each tryout has been simulated repeatedly five times (Run0, Run1, Run2, Run3 and Run4).

8.2 Particle Swarm Trajectories

▶ **Purpose:**

This first tryout shows how PSO algorithm works. The particle trajectories, the cost function and several errors will be analysed.

▶ **Results:**

An example of PSO run is depicted on figure 8.1. In these figures the particles trajectories are showed.

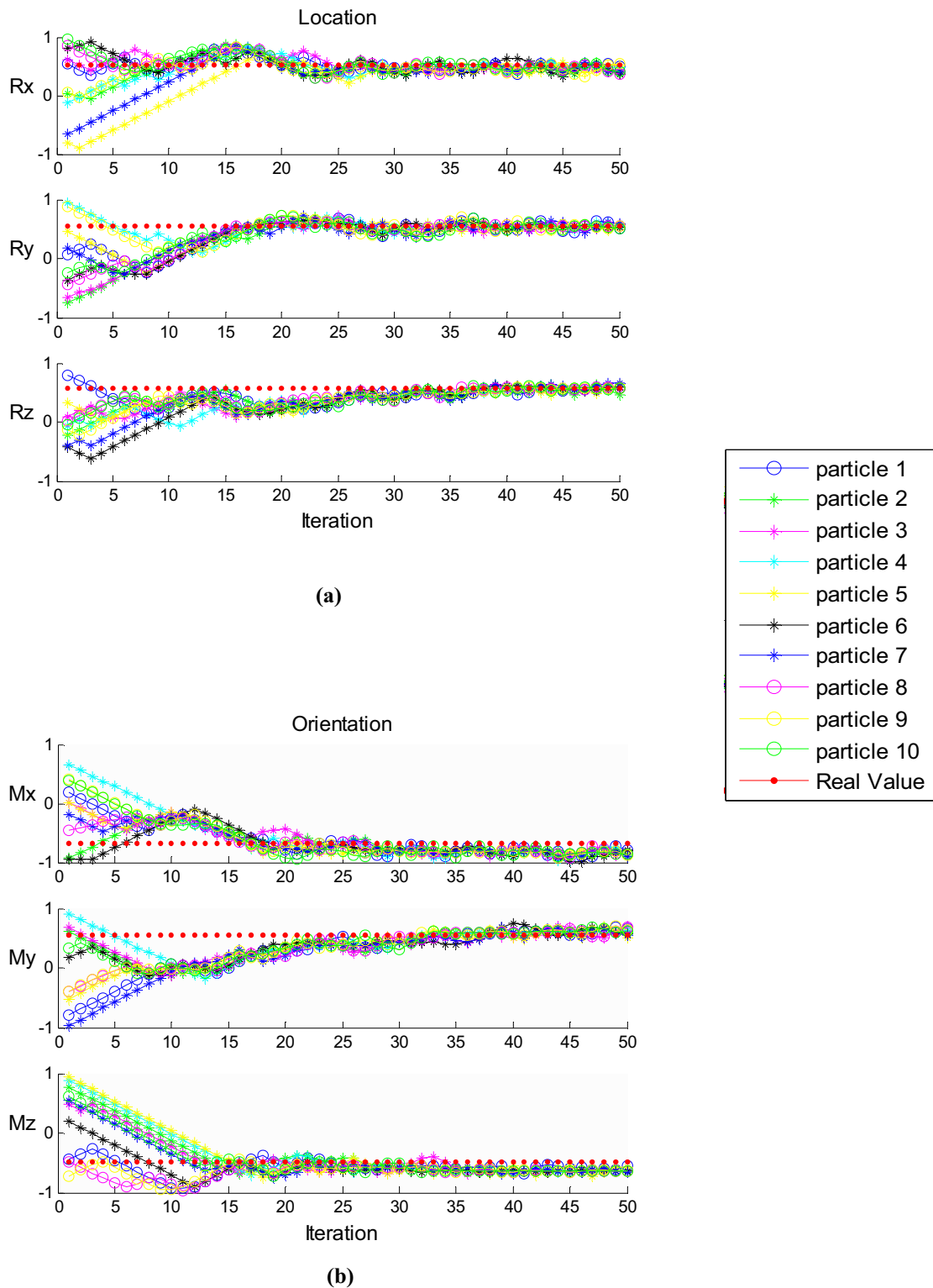


Figure 8.1. Example of PSO trajectories with 10 particles. **(a)**Dipole Location, **(b)** Dipole Orientation.

Ten dipoles start in the space randomly and their goal is to find the dipole location and orientation. In figure 8.1, after several iterations, all the particles eventually swarm over the optimum position.

In order to confirm this phenomenon, five separate trials have been analysed (Run0, Run1, Run2, Run3 and Run4). In the figure 8.2, the properly behaviour of the PSO in *Source Localization Problems* is verified. This figure depicts the minimum cost value (cost of gbest) for each run versus the iteration number. All the runs converge in the minimum cost, reaching the right location.

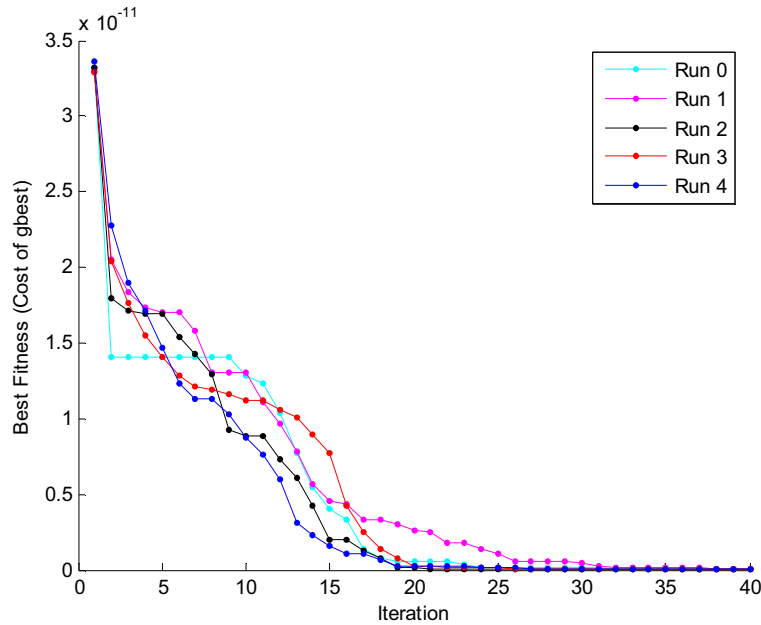


Figure 8.2. Fitness values versus number of iterations.

Going into detail, the difference between the exact values and the localized dipole locations and intensity has been analysed for each run (table 8.2).

	<i>Location</i>			<i>Moment [nA·m]</i>			I
	r_x	r_y	r_z	m_x	m_y	m_z	
Exact values	0.52	0.55	0.57	-0.68	0.55	-0.49	30
Run0	0.47	0.55	0.52	-0.70	0.54	-0.51	28.67
Run1	0.47	0.54	0.54	-0.83	0.64	-0.61	26.01
Run2	0.47	0.54	0.53	-0.99	0.75	-0.71	23.07
Run3	0.48	0.55	0.54	-0.96	0.76	-0.71	23.47
Run4	0.48	0.54	0.54	-0.70	0.53	-0.51	28.59
Mean	0.47	0.54	0.53	-0.68	0.52	-0.49	31.19
Standard Deviation	4.6e-2	7.7e-3	3.6e-2	7.7e-3	2.6e-2	7.7e-3	2.15

Table 8.2. Comparison between the localized dipoles results and the exact values.

The first impression is that the location and the dipole moment values are precise. Just a few millimetres in the dipole location and a few nA·m in de dipole moment.

This accuracy is also reflected by the root-mean-square (rms) error. In table 8.3 this average error has been calculated for the location and the moment parameters (\bar{e}_1 , \bar{e}_2 and \bar{e}_3).

$$\text{Exact Dipole Parameters} = (r_x, r_y, r_z, m_x, m_y, m_z, I)$$

$$\text{Simulated Dipole Parameters} = (\bar{r}_x, \bar{r}_y, \bar{r}_z, \bar{m}_x, \bar{m}_y, \bar{m}_z, \bar{I})$$

$$\bar{e}_1 = \sqrt{(m_x - \bar{m}_x)^2 + (m_y - \bar{m}_y)^2 + (m_z - \bar{m}_z)^2} \quad (8)$$

$$\bar{e}_2 = \sqrt{(r_x - \bar{r}_x)^2 + (r_y - \bar{r}_y)^2 + (r_z - \bar{r}_z)^2} \quad (9)$$

$$\bar{e}_3 = \left| \frac{I - \bar{I}}{I} \right| \quad (10)$$

Errors :	Location	Orientation	I
	\bar{e}_2	\bar{e}_1	\bar{e}_3
Run0	7.07e-2	3e-2	2.5e-2
Run1	6.48e-2	2.2e-2	5.13e-2
Run2	6.48e-2	2.4e-2	1e-2
Run3	5e-2	2.4e-2	1.06e-2
Run4	5.04e-2	3.3e-2	3.33e-2
Mean:	6.01e-2	2.66e-2	2.60e-2
Standar Deviation:	8.3e-3	4.1e-3	1.55e-3

Table 8.3. Errors: Location and Moment

Once analysed this example several tryouts with different source location and moment parameters were simulated. In all these cases PSO reached the goal with errors similar to the table 8.2 and 8.3.

About computational cost the whole process (five runs) takes around 4 min. Initially for each run is used 100 iterations although as it is showed in figure 8.2 each simulation needs around 35 iterations to converge towards the minimum cost.

► **Conclusión:**

PSO effectively finds the dipole parameters within optimum error range (millimetre errors). Besides, this algorithm also finds the dipole moment. It is not influential in the localization problem but is an important parameter. As well as having proper work behaviour, the computational cost is extraordinarily low.

In order to solve the dependency with the initial random points and try to improve the data accuracy, some ideas will be proposed in the *further research section*.

8.3 Swarm Size

► **Purpose:**

In this tryout it is showed the algorithm behaviour depending on the swarm size. It will be analysed four cases where the number of particles varies from 2, 5, 7 and 10.

► **Results:**

For each swarm size the cost function versus the iteration and the time required for reaching the minimum cost have been evaluated (figure 8.3).

The behaviour for 2 particles is poor. A large number of iterations are needed in order to reach good cost values. Curves for 5, 7 and 10 particles showed that PSO found successfully the minimum cost. With a swarm size of 10 particles it seems that the algorithm converges faster (20 iterations). However, as it is showed in the second figure, the computational cost is higher with 10 particles than with 7 particles. It is because the most computational time is used to evaluate the cost function. Thus, it takes longer for 10 particles (more cost function evaluations) than for 7 particles.

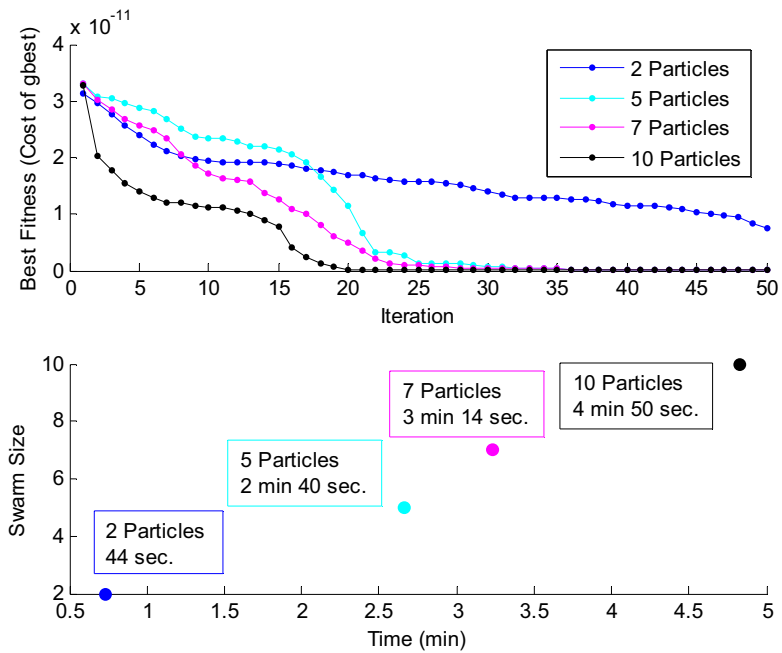


Figure 8.3. Best fitness and Computational time (2,5,7,10 particles).

In order to choose an optimum size, it should be analysed, in addition, the location accuracy. In figure 8.4 the location error dispersion has been depicted on for six different swarm sizes: 2, 5, 7, 10, 20 and 30 particles.

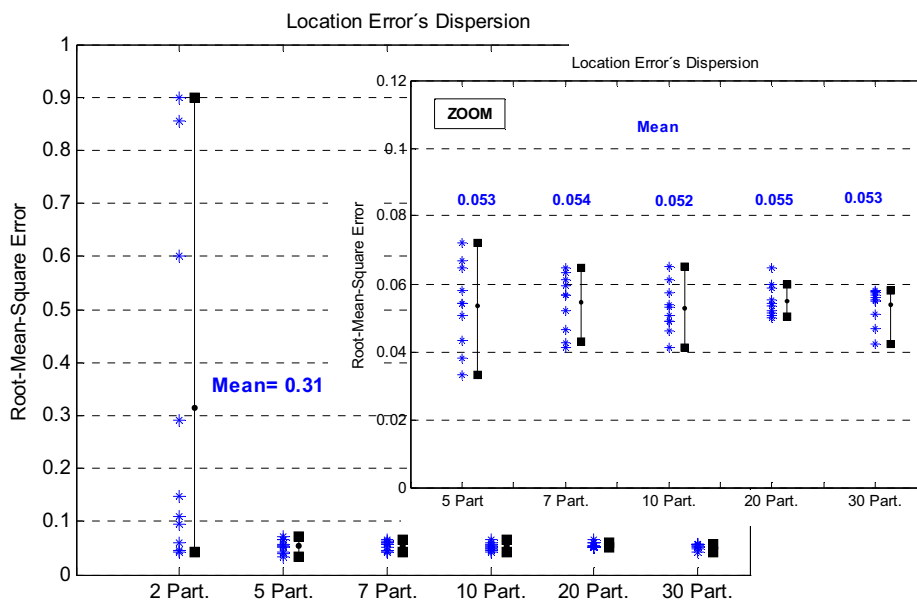


Figure 8.4. Location error dispersion for 2, 5, 7, 10, 20 and 30 particles (10 runs for each size).

As we can see, 2 particles is a bad choice due to its wide error range. The rest of the cases (5, 7, 10, 20 and 30 particles) work properly. Each one of them has similar average error that it is suitably small to get good results. The difference between them is the computational cost which reaches 19 min and 30 min with 20 and 30 particles respectively.

► **Conclusion**

In [44] Shi and Eberhart reported that “PSO is not sensitive to the population size”. In this case, we have found that it is generally true in terms of performance, but not in terms of computational cost.

A general size of 10 particles appears to be a good choice due to the accuracy of the localization results (average error 0.052) and the low computational cost (4 min 50 sec).

8.4 COGNITIVE / SOCIAL RATE

► **Purpose:**

To see the difference between the cognitive and social components, we have analysed two special cases.

One of them is called *Cognition-Only* model which consists of remove the social behaviour and move the particles only according to their own search history (zero for the social component in the velocity update equation (4)):

$$c_2 = 0$$

$$\vec{v}_i(t) = w \cdot \vec{v}_i(t-1) + c_1 \text{rand}(\cdot) \cdot (\vec{x}_{pbest\ i} - \vec{x}_i(t))$$

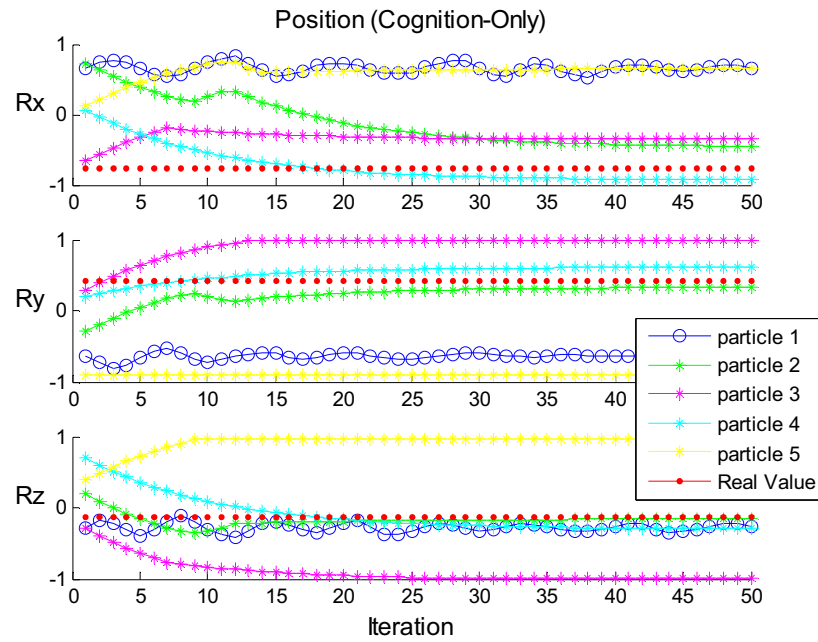
The other one is called *Social-Only* model. In this case, the particles move toward the one that found the lowest cost so far without the information about its own best position (zero for the cognitive component in the velocity update equation (4)):

$$c_1 = 0$$

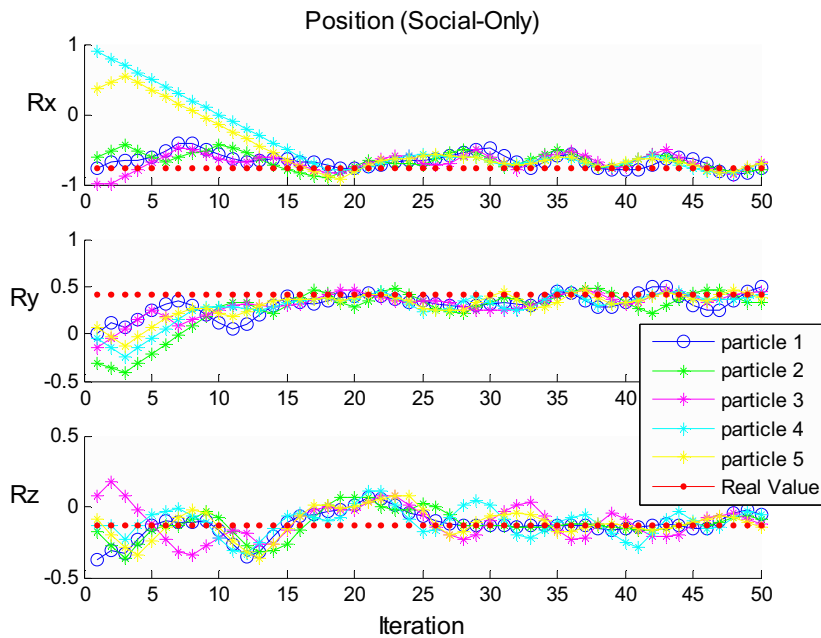
$$\vec{v}_i(t) = w \cdot \vec{v}_i(t-1) + c_2 \text{rand}(\cdot) \cdot (\vec{x}_{gbest} - \vec{x}_i(t))$$

► **Results:**

Figure 8.5 depicts on the particles position trajectories for the cognition-only and social-only models.



(a)



(b)

Figure 8.5. Dipole location trajectories (a) Cognition-Only (b) Social-Only

These trajectories show that the cognition-only model doesn't reach the true position. However, the social-only model has a good behaviour and tends to the goal position very fast. This behaviour is also depicted on figure 8.6 where the best fitness for each case is represented.

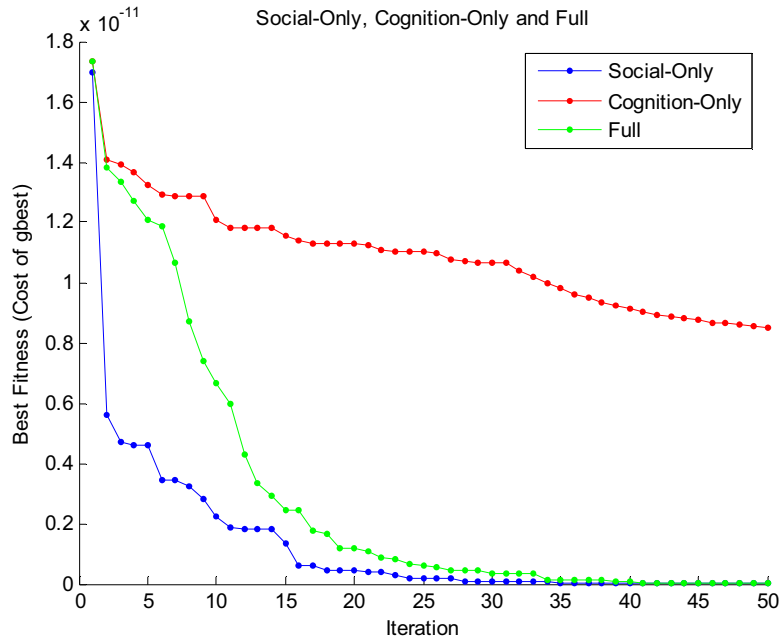


Figure 8.6. Best fitness returned by three PSO runs with different velocity update models: Social-Only, Cognition-Only and Full (setting the two components equal to 1.49).

As we can see, the cognition-only model doesn't find the minimum cost while the social-only and full models find the true minimum. If we compare the full model with the social model, we realize that the social one converges faster than the full one.

► **Conclusion:**

The cognitive-only model is more exposed to failure than the social model which is more efficient even comparing with the full model. However, the full model is more generally used in the literature. These results confirm recent studies like [39] [42] and as the sociobiologist E.O Wilson suggested, the social sharing of information offers an evolutionary advantage [43].

8.5 SNR (signal to noise ratio)

► **Purpose:**

Because of the “inverse crime” and because of EEG problems deal with very small signals, noisy environments should be taken into account. In order to study the noise dependency, white Gaussian noise was added to the electric field forward data. Five different noise levels—no noise and 20, 15, 10 and 5 dB (signal to noise ratio)—were simulated.

► **Results:**

Figure 8.7 shows the particle trajectories with respect to the various noise levels.

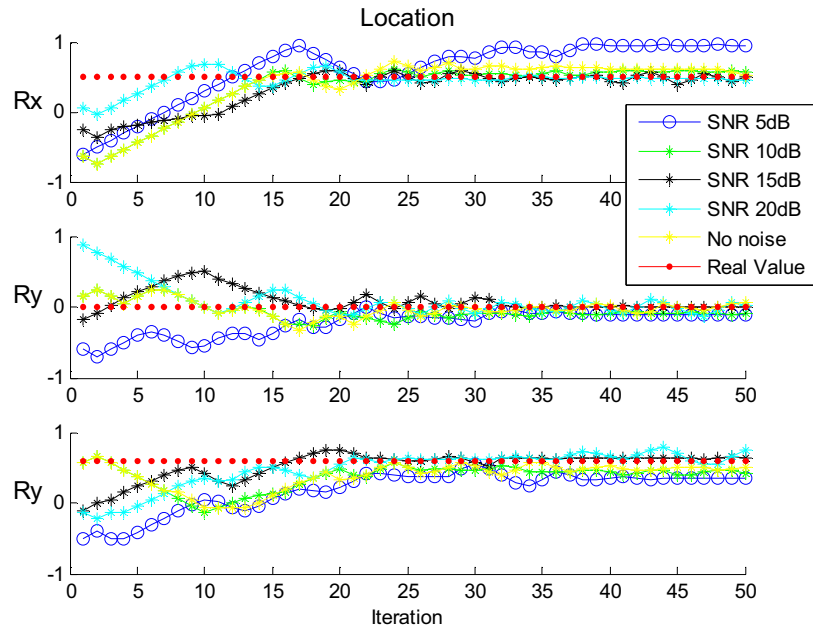


Figure 8.7. Particle trajectories for different SNR

As we can see the trajectories have worse behaviour when the SNR is too low, 5 dB and 10 dB. In both cases, particles don't find the true minimum due to they don't reach the minimum cost, figure 8.8.

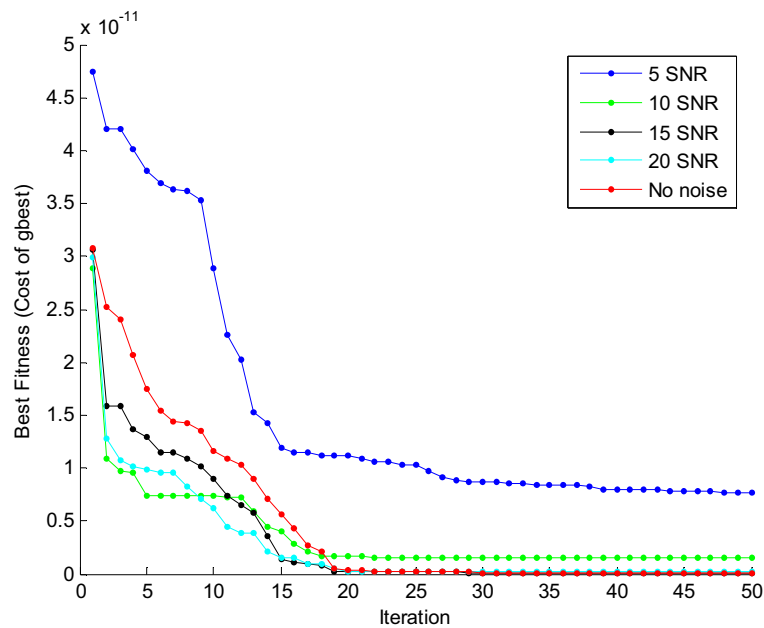


Figure 8.8. Best fitness (5 SNR, 10 SNR, 15 SNR, 20 SNR, No noise)

Figure 8.9 shows the location root mean square error dispersion and figure 8.10 shows the variation of the localization error according to the increment of noise levels.

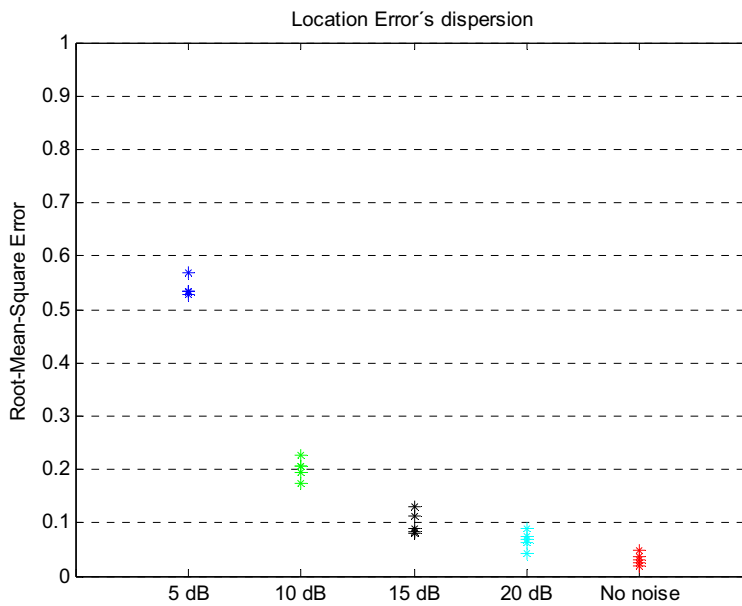


Figure 8.9. Location error dispersion (5 SNR,10 SNR, 15 SNR, 20 SNR, No noise).

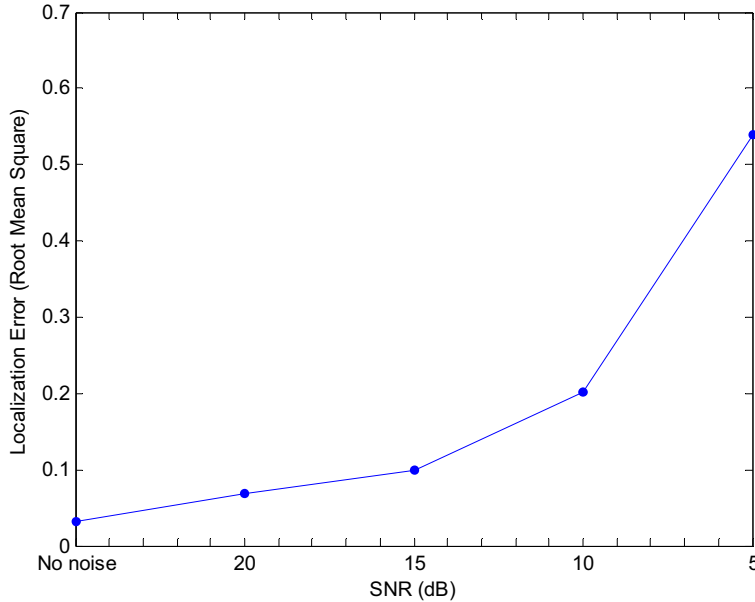


Figure 8.10. Localization error according to the increase of noise level. Localized errors are average values from 5 repeated simulations.

Both figures show the worsening of the solution precision when the noise increases. In case of having signals with SNR less than 15 dB, the localization error is too high to get good results.

► **Conclusion:**

The increment of noise decreases the solution accuracy, which means that the noisy or corrupted data make the localization problem more difficult. In case of noisy environments the damage is less significant with at least 15 dB of SNR.

Chapter 9

Real Case

In this tryout, real EEG measurement data were used. As in previous tryouts, the goal is to locate the source of a recorded EEG stimulus.

The EEG signals were provided by the *Department of Neuroscience and Physiology, Göteborg University, Sweden*. These EEG signals are the response of voluntary finger movements and they were reported in the paper [46] by Malin C.B. Aberg and Johan Wessberg.

Before analysing the PSO simulations results and in order to understand the following discussion, two important points are necessary introduced. In the first one, EEG Acquisition, the main concepts about the signal acquisition are explained (for more details [46]). The second point is related to the brain anatomy; only the most relevant concepts for this thesis are introduced.

9.1 EEG Acquisition

The study was performed with the participation of four untrained subjects. The subjects, seated in a chair, were instructed to move either the left or the right index finger in a brisk, according to cues presented on a screen. The period of the randomized cues was four seconds. Each cue was presented during three seconds. In this period the subject moved the finger at a determined point. Between 250-900 movements were registered for each subject.

EEG data were acquired at a sampling rate of 256Hz with 32 scalp electrodes positioned according to the extended 10/20 system. These data were pre-processed, and 100 epochs of one second before beginning the movement and 0.5 second after the movement were extracted.

9.2 Brain Concepts

9.2.1 Lobes of the Brain

The brain is divided into a right hemisphere and a left hemisphere. The hemispheres communicate with each other through a thick band of 200-250 million nerve fibres called the *corpus callosum*. Each hemisphere of the brain has four distinct sections, or lobes. These are the *frontal*, *parietal*, *temporal* and *occipital lobes* (figure 9.1).

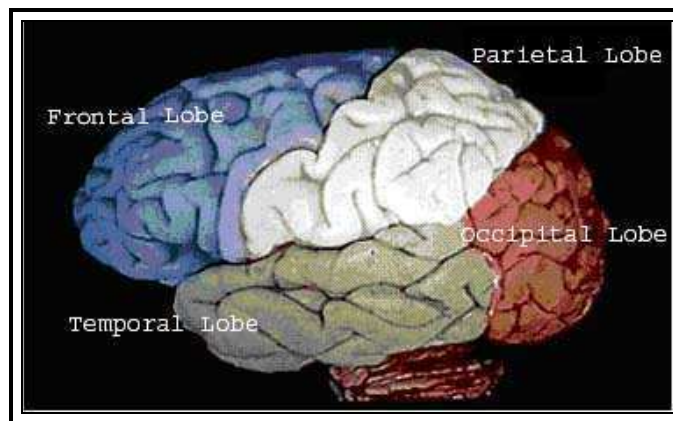


Figure 9.1. Brain Lobes

These lobes perform their own specific functions:

- **Frontal lobe:** concerned with reasoning, planning, parts of speech and movement (motor cortex), emotions, and problem-solving.
- **Parietal lobe:** concerned with perception of stimuli related to touch, pressure, temperature and pain.
- **Temporal lobe:** concerned with perception and recognition of auditory stimuli and memory.
- **Occipital lobe:** concerned with many aspects of vision.

9.2.2 Laterality discrimination

The lobes in each hemisphere control the opposite side of the body. In general, sensory information from the left side of the body crosses over to the right side of the brain and information from the right side of the body crosses over to the left side of the brain.

9.3 Procedure description

Several researches about finger movement classification [46] are focused on a few central electrodes locations like C3 and C4. It is because these electrodes are closed to the left and right-hand *primary motor cortex*, concerned with the initiation of voluntary movement and the *somatosensory cortex*, which receives tactile information from the body (figure 9.2).

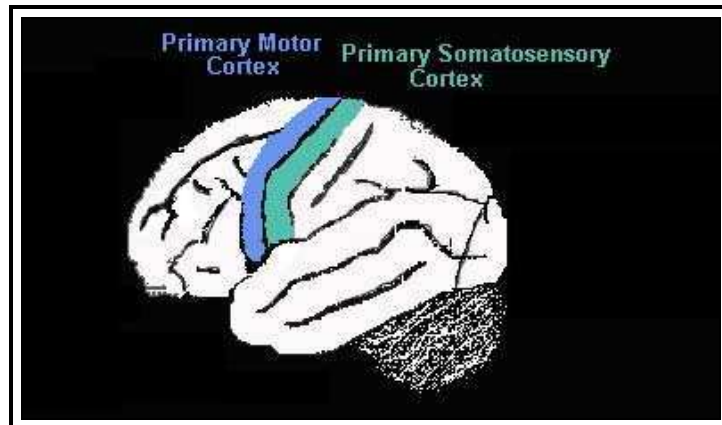


Figure 9.2. Primary Motor Cortex and Primary Somatosensory Cortex.

Figure 9.3 shows the EEG signals recorded by C4 and C3 electrodes. These signals are the average of 416 movements [46].

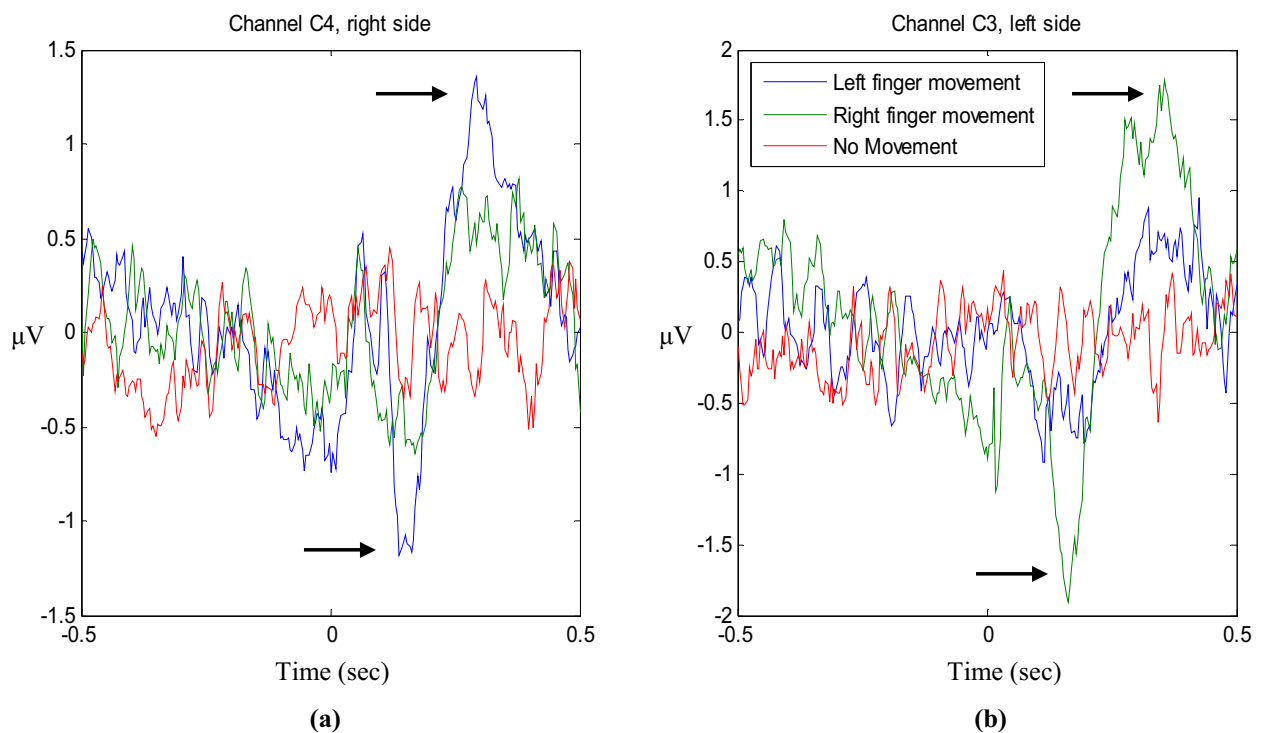


Figure 9.3. EEG signal. (a) Electrode C4 (b) Electrode C3.

Let us focus on the main peaks for each different movement and electrode. These signals confirm that the right side of the brain (C4 electrode) controls information from the left side of the body (left finger movement) and the left side of the brain (C3 electrode) controls information from the right side of the body (right finger movement).

Due to PSO algorithm works with discrete values, the EEG signal for each electrode should be previously sampled. In the right finger case, the samples have been selected around the maximum and minimum peaks in C3 electrode. On the other hand, in left finger case, the samples were chosen around the maximum and minimum peaks in C4 electrode.

Settings for PSO algorithm were listed in Table 8.1.

9.4 Results

After introducing all the potential samples in the PSO algorithm, the source location results were represented with the DipoleSimulator graphical interface (figure 9.4 and figure 9.5).

► Right Finger Movement:

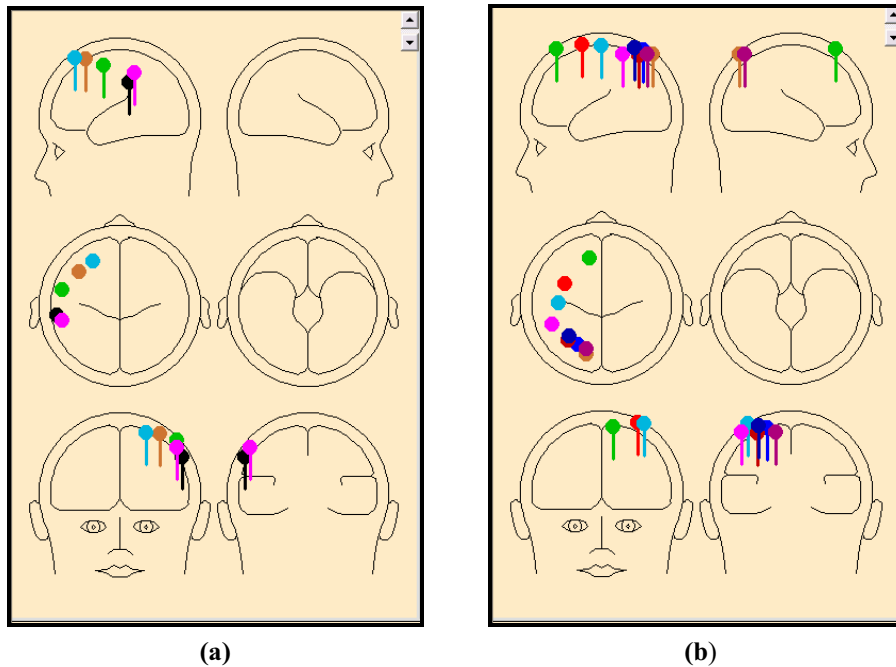


Figure 9.4. Right Source Location (a) 5 samples around minimum peak (b) 10 samples around maximum peak.

► **Left Finger Movement :**

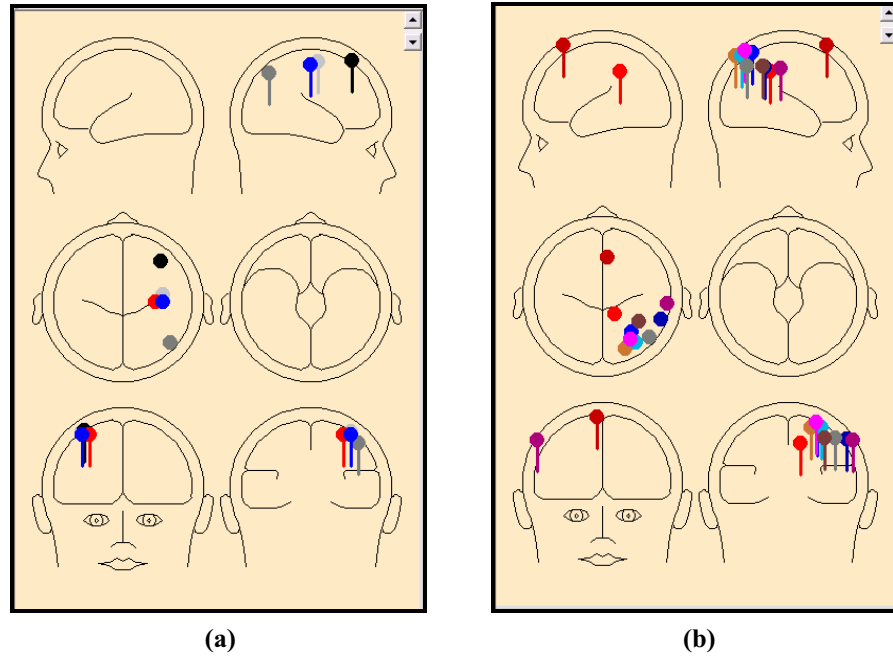


Figure 9.5. Left Source Location (a) Minimum Peak, 5 samples (b) Maximum Peak, 10 samples.

9.5 Discussion

The laterally discrimination is demonstrated in both cases. As figure 9.4 and figure 9.5 show, all the dipole results are localized in the hemisphere opposite to the finger movement side.

The minimum peak valley corresponds with the movement beginning (figure 9.4.a, figure 9.5.a). The results in this case, are localized in different points within the brain with any concrete pattern. Nevertheless, at the end of the movement (figure 9.4.b, figure 9.5.b) most of the source locations fit in the same place, in the parietal lobe. This lobe includes sensory areas responsible for feelings of temperature, touch, pressure and pain from the skin.

During the course of a finger movement, different areas with different dipole orientations are activated, and it is therefore difficult to find a unique source location. In order to provide most useful information about the PSO algorithm accuracy, EEG signals which involve less activated areas should be tested. In *further research* section will be suggested some EEG signals to be tested.

Chapter 10

Conclusions

This chapter analyzes the work done during this thesis, extracting the main conclusions both in the technical side and in the personal one.

10.1 Technical work and conclusions

Improved EEG analysis software will enable a faster and more precise diagnosis of the non-invasive EEG in the near future. Based on this idea, the goal of this thesis was to develop a method for transforming the scalp EEG into a source localization associated with the generated brain region

For doing so, a long period of reading of the relevant literature in the field was carried out. Once the state of the art was acquired, the first step consisted of designing the head and source models. We proposed the four layers sphere model and equivalent current dipoles. It was showed that using simple spherical models general information about source localization can be acquired. Furthermore, the most fundamental advantage of the ECD (equivalent current dipole) method is that it is very easy to implement and does not require any anatomical data of a human brain.

In the second step we calculated the scalp potential choosing before a properly electrode system. The potential surface topography can be reconstructed even when only a limited number of electrodes are available.

Once the forward problem solved, the next part consisted of solving the EEG inverse problem which is a nonlinear problem. In order to solve it a global optimization method called Particle Swarm Optimization (PSO) was proposed due to its simplicity in the algorithm. As it was explained the main difficulty of this method was that PSO parameters depend highly on the optimization problem environment.

Our results have proved that, Particle Swarm optimization can be used for optimizing the EEG inverse problem. In general, the magnitude of the localization error caused by using a spherical head model varies with the location of the source in the brain and the sphere parameters. However, PSO effectively finds the dipole localization with millimetres error magnitude. *Once again nature has provided us with a technique for processing information that is at once elegant and versatile* [32].

Using real EEG signals, this accuracy is not demonstrated in this thesis. As it was explained EEG signals which involve less activates areas than the finger movement does, should be used in order to provide this information.

10.2 Personal conclusions

My personal evaluation of this work is exceptionally good. The work done was state-of-the-art research, allowing me to introduce myself in an amazing field. I would like to underline that this project started with any knowledge about the biomedical field it deals with. It was a fascinating challenge that I accepted because the wish of discovering this new work area.

There were two basic conditions which make this thesis special:

CHALMERS. The fact of developing the thesis in the “Biomedical Engineering Division” of Chalmers University (Göteborg) was a pleasure. This department has different research groups that have received significant recognition in their research areas. My work environment was also connected to the Shalgrenska University Hospital (Göteborg). Several meetings with a neurological doctor took place during the project development. It helped me to inspire my work at the same time that it incited me.

SWEDEN. This thesis has been carried out in Göteborg (Sweden) during my Erasmus year. I have had the opportunity to discover this country, enjoying its amazing nature and sharing its culture. Furthermore, during this period I have known lot of new people from different countries which helped me to discover others ways of life. Undoubtedly I advise this learning experience.

Chapter 11

Further Research

This chapter pretends to guide readers interested in continuing and extending thin work. As in any research project, the work that has been presented in this thesis can be both improved and extended.

11.1 Future research lines

► Real Head

As I said if only general information about a source is needed then a simple spherical model can be used. Greater localization accuracy requires more realistic and complicate head models. Due to EEG is highly affected by the inhomogeneity of human brain structures [47], many studies on the realistic head model have been performed like *boundary element method* (BEM) and *finite element method* (FEM).

In recent years, BEM and FEM have been applied to localize electrical activity in the brain using information of the head structure from *magnetic resonance imaging* (MRI) and *computed tomography* (CT). These methods are useful for quantitative analysis of brain functions.

► Electrodes models

The international 10-20 electrode system used in the EEG analysis is capable of analyzing epilepsy or an evoked potential. The number of electrodes must be increased to improve the resolution. Therefore, 21, 49 or 91 electrodes systems are proposed for future researches.

▶ **Multiple Dipoles**

In this thesis a single dipole was used but in realistic EEG cases the current sources within the brain may not be accurately described by a single current dipole. For this reason, electrical activity must be approximated by multiple dipoles to achieve a more realistic analysis method. Different sources may be active simultaneously and the location and orientation may also vary. As it was explained, the head can be modelled as a linear volume conductor which permits the use of the superposition principle. Therefore, the potential response due to multiple sources may be approximated by combining the individual potentials of a group of dipoles at different times, orientations, and location.

However, a current dipole has many elements, and if the number of dipoles is increases, it becomes more difficult to realize a solution. To solve this problem, it is suggested to employ a model that considers the distribution of the electrical activity and brain structure. The idea is to localize the electrical activity by focussing on anatomical features restricting the evoked potential to appropriate sensory and association areas of the cerebral cortex such as motor, visual and auditory cortexes.

▶ **MEG**

Contrary to the electroencephalography, the MEG is not highly affected by the inhomogeneity of human brain structures [47]. Recent studies showed that the spherical model could substitute for a realistic head model sufficiently [48]. The suggestion consists of solving the MEG inverse problem using the spherical four layers method and the PSO method.

▶ **Noise**

In order to study the noise dependency a white Gaussian noise was added to the simulated signal. However, other method more realistic can be used. It consists of adding in specific proportion, for each electrode, the contributions from many sources (for example 100) to the dipole synthetic data. This method is used in BESA.

▶ **Improvements on PSO**

The PSO algorithm is worth further developments for using in EEG dipoles modelling. The future work can involve the next points:

- Since PSO algorithm starts with random localization, each tryout was simulated several times. In order to solve the dependency with the initial random points and try to improve the result precision, the suggestion is start with concrete dipole localizations. Two possible distributions are proposed: the first one is based on having a priori knowledge that the sensory activity is restricted to sensory areas of the cortex (the initial dipoles will be localized in these areas). The second distributions consist of localizing the dipoles uniformly within the sphere, for example following a compact hexagonal placement.

- In case of using real head models or incrementing the number of electrodes, the computational cost will be increase. In order to avoid it, the next idea is proposed. Let us start running the algorithm with 30 or more initial dipoles. As the iterations are taking place the dipoles with the highest cost value will be removed for the next iteration, and so on. By this way, the algorithm could become more efficiently improving the computational cost.
- Extending the PSO implementation to a parallel computing architecture. As it was explained, in this thesis, the particles (dipoles) are treated sequentially in the cost function evaluation. This task could be done in parallel if a multi-processors machine is available [49].

► Real EEG Singals

To analyze the accuracy of the proposed method, particular EEG signals which involve less activated brain areas should be tested. These signals are called *Sensory Evoked Potential (SEP)* and are recorded from the central nervous system following stimulation of sense organs. The usual sites for SEP stimulation are the median nerve at the wrist, the common peroneal nerve at the knee, and the posterior tibial nerve. A SEP also may be recorded by stimulating the skin in various dermatomal areas, but the response is much weaker.

► Sources Tracking

Once the accuracy is tested, the next step could consist of tracking source movements. In this process the signal should be adequately sampled.

References

- [1] P.L Nunez, “Electric fields of the brain”, New York: Oxford 1981.
- [2] C.D. Tesche, J.Karhu, “THETA oscillations index human hippocampal activation during a working memory task”, *Proc.Natl.Acad.Sci. USA*, vol 97, pp.919-924, 2000.
- [3] C.D. Tesche, J.Karhu, “Somatosensory evoked magnetic fields arising from sources in the human cerebellum”, *Brain Res.*, vol. 744, pp.23-31, 1997.
- [4] C.E. Tenke, C.E. Schroeder, J.C. Arezzo, and H.G. Vaughan, “Interpretation of high-resolution current source density profiles: A simulation of sublaminar contributions to the visual evoked potential”, *Exp.Brain Res.*, vol. 94, pp.183-192, 1999.
- [5] R.R. Llinas, U. Ribary, D. Jeanmonod, E. Kronberg, P.P Mitra, “Thalamocortical dysrhythmia: A neurological and neuropsychiatric syndrome characterized by magnetoencephalography”, *Proc.Natl.Acad.Sci. USA*, vol 96, pp.15222-15227, 1999.
- [6] Cuffin BN, “A Method for localizing EEG sources in realistic head models”, *IEEE Trans Biomed Eng.*, vol.BME-42, No. 1, 1995.
- [7] Ary JP, Klein SA, Fender DH, “Location of sources of evoked potential: Corrections for skull and scalp thicknesses”, *IEEE Trans Biomed Eng.*, vol.BME-28, NO. 6,1981.
- [8] Stok, CJ, “The influence of model parameters on EEG/MEG single dipole source estimates”, *IEEE Trans Biomed Eng.*, vol.BME-34, pp.289-296, 1987.
- [9] Cuffin BN, “Effects of local variations in skull and scalp thickness on EEG’s and MEG’s”, *IEEE Trans Biomed Eng.*, vol.BME-40, pp.42-48, 1993.
- [10] J.Pascau, P.Rojo, A.Santos, M.A.Pozo, M.Desco, “Localización especial de dipolos de electroencefalografía en imágenes de resonancia magnética”.
- [11] Crouzeix Anne, “Étude de méthodes permettant de localiser les sources de courant intracérébrales à partir d’ enregistrements couplés des potentiels électriques et des champs magnétiques sur le scalp de l’homme”, *Processus mentaux et activation cérébrale*, Lyon,1997

- [12] Program Brain Electric Source Analysis. BESA v.2.2 MEGIS Software GmbH, Munich, Alemania
- [13] G.E. Chatrian, E. Lettich and P.L. Nelson, “Ten percent electrode system for topographic studies of spontaneous and evoked EEG activity”, *Am. J. EEG Technol.* 25, pp. 83–92, 1985.
- [14] Robert Oostenveld, “Improving EEG Source Analysis using Prior Knowledge”, Proefschrift Katholieke Universiteit Nijmegen, 2003.
- [15] Jaakko Malmivuo & Robert Plonsey: *Bioelectromagnetism - Principles and Applications of Bioelectric and Biomagnetic Fields*, Oxford University Press, New York, 1995. <http://butler.cc.tut.fi/~malmivuo/bem/bembook/>
- [16] http://www.besa.de/index_home.htm
- [17] <http://www.neuroscan.com/landing.cfm>
- [18] F.N Wilson and R.H Bayley, “The electric field of an eccentric dipole in a homogeneous spherical conducting medium”, *Circulation*, vol.1, pp.84-92, 1950.
- [19] S.Rush, D.A. Driscoll, “Current distribution in the brain from surface electrodes”, *Anesth. Analg.*, vol. 47, pp.717-723, 1968.
- [20] B.N.Cuffin, D.Cohen, “Comparison of the magnetoencephalogram and electroencephalogram”, *Electroencephalogr. Clin. Neurophysiol.*, vol.47, pp.132-146, 1979.
- [21] C.J. Stock, “The inverse problem in EEG and Meg with application to visual evoked responses”, in *CIP Gegevens Koninklijke Bibliotheek*. The Hague, The Netherlands, 1986.
- [22] J.C. de Munk, “The potential distribution in a layered anisotropic spheroidal volume conductor”, *J. Appl. Phys.*, vol. 64, pp.464-470, 1988.
- [23] Mingui Sun, “An efficient algorithm for computing multishell spherical volume conductor models in EEG dipole source localization”, *IEEE Trans Biomed Eng.*, vol.BME-44, No. 12, 1997.
- [24] John C.Mosher, Richard M. Leahy, Paul s.Lewis, “EEG and MEG: Forward solutions for inverse problem”, *IEEE Trans Biomed Eng.*, vol. 46, No. 3, 1999.
- [25] Yehuda Salu, Leonardo G. Cohen, Douglas Rose, Susumu Sato, Conrad Kufta, Mark Hallett, “An improved method for localizing electric brain dipoles”, *IEEE Trans Biomed Eng.*, vol. 37, No. 7, 1990.
- [20] P.Berg, M. Scherg, “A fast method for forward computation of multiple-shell spherical head models”, *Electroenceph. Clin. Neurophysiol.*, vol. 90, pp. 58-64, 1994.
- [27] J.C de Munk, M.J Peters, “A fast method to compute the potential in the multisphere model”, *IEEE Trans Biomed Eng.*, vol. 40, pp. 1166-1174, 1993.

- [28] Z.Zhang, “A fast method to compute surface potentials generated by dipoles within multilayer anisotropic spheres”, *Phys. Med. Biol.*, vol. 40, pp.335-349, 1995.
- [29] Christoph M. Michael, Micah M.Murray, Göran Lantz, Sara Gonzalez, Laurent Spinelli, Rolando Grave de Peralta, “EEG source imaging”, *International Federation of Clinical Neurophysiology, Published by Elsevier*,2004.
- [30] Sylvain Baillet, John C. Mosher, and Richard M. Leahy, “Electromagnetic Brain Mapping”, *IEEE Signal and Processing Magazine*, 2001.
- [31] J. Kennedy and R.C. Eberhart, “Swarm Intelligence”, Morgan Kaufmann Publishers, 2001. <http://www.engr.iupui.edu/~eberhart/web/PSObook.html>
- [32] J.Kennedy and R.C Eberhart, “Particle swarm optimization”, in *Proc. IEEE Conf. Neural Networks IV*, Piscataway, NJ, 1995.
- [33] Jacob Robinson and Yahya Rahmat-Samii, “Particle Swarm Optimization in Electromagnetics”, *IEEE Trans. Antennas Propagat.*, vol. 52, no.2, pp. 397-407, Feb.2004.
- [34] Y.Shi, R.C. Eberhart, “Parameter selection in particle swarm optimization”, *Proceedings of the IEEE Congress on Evolutionary Computation*, IEEE Press, pp. 1945-1950, 1999.
- [35] A.Carlisle, G.Doizer, “An off- the-shelf PSO”, *ProcWorkshop Particle Swarm Optimization*, Indianapolis, IN, 2001.
- [36] R.C.Eberhart, Y.Shi, “Particle swarm optimization: Developments, applications and resources”, *Proceedings of the IEEE Congress on Evolutionary Computation*, *IEEE Press*, Seoul, Korea, 2001.
- [37] Y. Shi, R.C. Eberhart, “A modified particle swarm optimizer”, *Proceedings of the IEEE Congress on Evolutionary Computation*, Piscataway, USA, pp. 69–73,1998.
- [38] R.C. Eberhart, Y.Shi, “Comparing inertia weights and constriction factors in particle swarm optimization”, *Proceedings of the IEEE Congress on Evolutionary Computation*, San Diego, USA, pp. 84–88, 2000.
- [39] Anthony Carlisle, Gerry Dozier, “An Off-The-Shelf PSO”, *Proceedings of the Workshop on Particle Swarm Optimization*, Huntingdon College, Auburn University, 2001.
- [40] F.van den Bergh, A.P. Engelbrecht, “A study of particle swarm optimization particle trajectories”, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2005.
- [41] E.S. Peer, F. van den Bergh, A.P. Engelbrecht, “Using Neighbourhoods with the Guaranteed Convergence PSO”, Department of Computer Science, University of Pretoria, Pretoria, South Africa.

- [42] J. Kennedy, “The Particle Swarm: Social Adaptation of Knowledge”, in *proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 303-308, Apr.1997.
- [43] Wilson, E.O. “Sociobiology: The new synthesis”. *Belknap Press, Cambridge, MA*. 1975.
- [44] Shi, Y. and Eberhart, R.C, “Empirical Study of Particle Swarm Optimization”, *Congress on Evolutionary Computing*, vol. III, pp. 1945-1964, 1999.
- [45] Armand Wirgin, “The inverse crime”, arXiv:math-ph/0401050 v1 , 2006.
http://arxiv.org/PS_cache/math-ph/pdf/0401/0401050v1.pdf
- [46] Malin C.B. Aberg, Johan Wessberg, “Evolutionary optimization of classifiers and features for single trial EEG Discrimination”, *BioMedical Engineering OnLine* 2007.
- [47] M.S. Hämäläinen et al., “Magnetoencephalography. Theory, instrumentation and applications to the noninvasive study of human brain function”, *Rev. Mod. Phys.*, vol. 65, pp.413-497, 1993.
- [48] R. V. Uitert and C.Johnson, “Can a spherical model substitute for a realistic head model in forward and inverse MEG simulations?”, in *Proc. Conf. Biomagnetism*, Jena, Bermany, 2002.
- [49] J.F. Schutte, J.A. Reinbolt, B.J Fregly, R.T. Haftka, and A.D. George, “Parallel Global Optimization with the Particle Swarm Algorithm”, in *International Journal For Numerical Methods in Engineering*, pp. 2296-2315, June 2004.
- [50] R.M. Arthur and D.B. Geselowitz, “Effects of inhomogeneities on the apparent location and magnitude of a cardiac current dipole generator”, *IEEE Trans Biomed Eng.*,vol. BME-17, pp. 141-146, Apr.1970.

Images References

- Figure 2.1 Jaakko Malmivuo & Robert Plonsey: *Bioelectromagnetism – Principles and Applications of Bioelectric and Biomagnetic Fields*, Oxford University Press, New York, 1995.
- Figure 2.2 Jaakko Malmivuo & Robert Plonsey: *Bioelectromagnetism – Principles and Applications of Bioelectric and Biomagnetic Fields*, Oxford University Press, New York, 1995.
- Figure 3.1.a B.Neil Cuffin, “EEG Dipole Source Localization”, *IEEE Engineering in medicine and biology*,1998.
- Figure 3.1.b http://www.nlm.nih.gov/research/visible/vhp_conf/krabbel/krabbel.htm
- Figure 3.3 <http://www.cephalon.dk/WaveGuardCap.htm>
- Figure 3.4 Jaakko Malmivuo & Robert Plonsey: *Bioelectromagnetism – Principles and Applications of Bioelectric and Biomagnetic Fields*, Oxford University Press, New York, 1995.
- Figure 3.5 Jaakko Malmivuo & Robert Plonsey: *Bioelectromagnetism – Principles and Applications of Bioelectric and Biomagnetic Fields*, Oxford University Press, New York, 1995.
- Figure 6.1 Jacob Robinson and Yahya Rahmat-Samii, “Particle Swarm Optimization in electromagnetics”, *IEEE Trans. Antennas Propagat.*, vol. 52, no.2, pp.1945-1950, 2004.
- Figure 6.2 Jacob Robinson and Yahya Rahmat-Samii, “Particle Swarm Optimization in electromagnetics”, *IEEE Trans. Antennas Propagat.*, vol. 52, no.2, pp.1945-1950, 2004.
- Figure 6.4 Jacob Robinson and Yahya Rahmat-Samii, “Particle Swarm Optimization in electromagnetics”, *IEEE Trans. Antennas Propagat.*, vol. 52, no.2, pp.1945-1950, 2004.

Figure 9.1 <http://faculty.washington.edu/chudler/lobe.html>

Appendix A

Forward Problem

In this appendix it is explained how to solve the forward problem. Let us start calculating the solution for the potential at the surface of a homogeneous sphere due to a dipole inside the sphere.

► Homogeneous Sphere:

The coordinate system is showed in figure A.1.

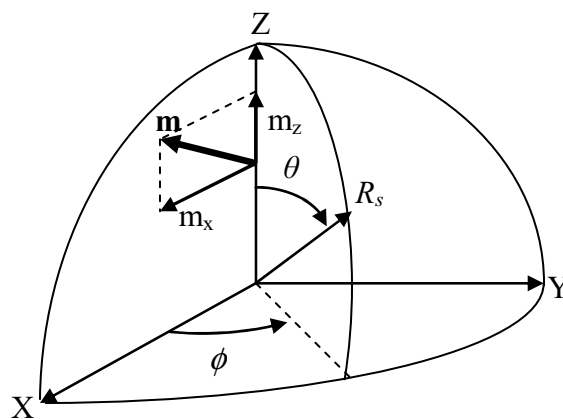


Figure A.1. Coordinate system for dipoles in spherical head model.

The electric potential at the outer surface is giving by the general solution of Laplace's equation $\Delta V=0$ and the particular solution of the dipole: $V=V_{\text{part}}+V_{\text{gen}}$, taking the boundary conditions into account.

$$V_{part} = \frac{1}{4\pi\sigma} \frac{\vec{m} \cdot (\vec{r} - \vec{b})}{|\vec{r} - \vec{b}|^3} \quad (\text{A.1})$$

Where b is the distance between the dipole (with moment m) and the centre of the sphere, r is the field point and σ the conductivity of the region where the dipole is situated.

The boundary conditions for this problem are:

- (1) V is continuous at $r = R_s$ (outer surface of the sphere).
- (2) The normal component of j is continuous at $r = R_s$, therefore

$$\left(\frac{\partial V}{\partial r} \right)_{r=R_s} = 0$$

- (3) The potential for r goes to zero should be finite.

Without loss of generality we can choose a coordinate system such that the dipole is at the Z -axis. As an example, we calculate the solution for a radial source m_z within a homogeneous sphere. In this case the potential distribution has axial symmetry and therefore is independent of the azimuthal angle, φ . In this case the general solution of Laplace's equation is:

$$V_{gen} = \sum_{n=0}^{\infty} (A_n r^n + B_n r^{-(n+1)}) P_n(\cos \theta) \quad (\text{A.2})$$

where P_n are the Legendre polynomials.

The particular solution expressed in spherical harmonic functions reads:

$$V_{part} = \frac{m_z}{4\pi\sigma} \frac{(r \cos \theta - b)}{|\vec{r} - \vec{b}|^3} = \frac{m_z}{4\pi\sigma} \frac{1}{rb} \sum_{n=0}^{\infty} n \left(\frac{b}{r} \right)^n P_n(\cos \theta) \quad (\text{A.3})$$

The total solution is the sum of the general and the particular solution (equation A.2 and A.3). Taking boundary condition 3 into account it becomes:

$$V = \sum_{n=0}^{\infty} A_n r^n P_n(\cos \theta) + \frac{m_z}{4\pi\sigma} \frac{1}{rb} \sum_{n'=0}^{\infty} n' \left(\frac{b}{r} \right)^{n'} P_{n'}(\cos \theta) \quad (\text{A.4})$$

From boundary condition 3 it follows that $B_n=0$ for all n .

Applying boundary condition 2 yields:

$$\begin{aligned} \frac{\partial V}{\partial r} &= \sum_{n=0}^{\infty} n A_n r^{n-1} P_n(\cos \theta) - \sum_{n'=0}^{\infty} \frac{m_z}{4\pi\sigma} n' (n'+1) \frac{b^{n'-1}}{r^{n'+2}} P_{n'}(\cos \theta) = 0 \\ A_n &= \frac{m_z}{4\pi\sigma} (n+1) \frac{b^{n-1}}{r^{2n+1}} \end{aligned} \quad (\text{A.5})$$

Combining equation A.5 and A.4 and taking $r = R_s$, the solution is:

$$V = \frac{m_z}{4\pi\sigma} \sum_{n=0}^{\infty} (2n+1) \frac{b^{n-1}}{R_s^{n+1}} P_n(\cos\theta) \quad (\text{A.6})$$

We can do the same for a dipole oriented in the X-direction. The solution reads:

$$V = \frac{m_x \cos\phi}{4\pi\sigma} \sum_{n=1}^{\infty} \left(2 + \frac{1}{n}\right) \frac{b^{n-1}}{R_s^{n+1}} P_n^1(\cos\theta) \quad (\text{A.7})$$

Where P_n^1 is the associated Legendre polynomial. The solution for a dipole in the Y-direction is analogous to the one in the X-direction.

The total solution for the potential at the surface of a homogeneous sphere due to a dipole inside the sphere on the Z-axis, with dipole moment $\mathbf{m}=(m_x, m_y, m_z)$, is:

$$V(\theta, \phi) = \frac{m_z}{4\pi\sigma R_s^2} \sum_{n=1}^{\infty} (2n+1) \left(\frac{b}{R_s}\right)^{n-1} P_n(\cos\theta) + \frac{m_x \cos\phi + m_y \sin\theta}{4\pi\sigma R_s^2} \sum_{n=1}^{\infty} \left(\frac{2n+1}{n}\right) \left(\frac{b}{R_s}\right)^{n-1} P_n^1(\cos\theta) \quad (\text{A.8})$$

► Inhomogeneous Shell Sphere:

The head is models as a sphere with four concentric-shells of volume conductors (figure A.2).

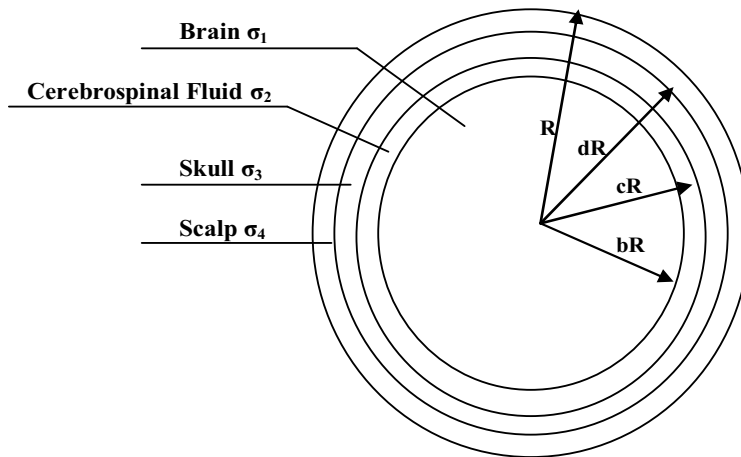


Figure A.2. Four concentric-shells head model

The dipole current source within the head model is defined by six parameters: *Dipole location vector*, \mathbf{r} (r_x, r_y, r_z) and *Current dipole moments*, \mathbf{m} (m_x, m_y, m_z) (figure A.3).

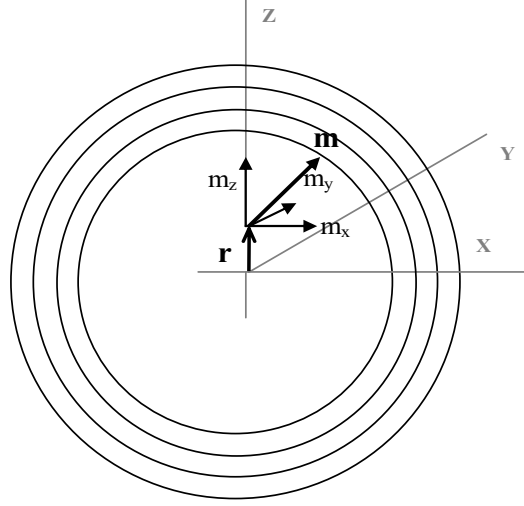


Figure A.3. Dipole within the four concentric-shells model. The centre of the sphere is at the origin.

Applying the method originally developed by Arthur and Geselowitz [50], a *correction factor* c_n is used for transforming homogeneous model source parameters to inhomogeneous shell model parameters.

If we considered a dipole situated at r , the potential value v at the surface point s (s_x, s_y, s_z) will be given by [25]

$$v = \frac{1}{4\pi\sigma_4 R^2} \sum_{n=1}^{\infty} c_n f^{n-1} m \cdot \left[r_0 P_n(\cos \theta) + t_0 \frac{P_n^1(\cos \theta)}{n} \right] \quad (\text{A.9})$$

where

- The dot after m denotes a dot product (m, r_0, t_0 are vectors).
- r_0 and t_0 are the radial and tangential unit vectors which depend on r and s but are independent of index n .
- σ_4 is the conductivity value for the scalp.
- $R = \sqrt{s_x^2 + s_y^2 + s_z^2}$ is the outer radius of the head.
- $f = \frac{|r|}{R}$ is the eccentricity of the dipole.
- θ denotes the angle between r and s .
- $P_n(\cos \theta)$ and $P_n^1(\cos \theta)$ are respectively the Legendre and associated Legendre polynomials of degree n .
- c_n , *correction factor*, represented a series determined by the model geometry and conductivity values.

The expression for c_n is given by [20][50]

$$c_n = \frac{(2n+1)^4 (cd)^{2n+1}}{\Gamma} \quad (\text{A.10})$$

with

$$\begin{aligned} \Gamma = & d^{2n+1} \left[b^{2n+1} n \left(\frac{\sigma_1}{\sigma_2} - 1 \right) \left(\frac{\sigma_2}{\sigma_3} - 1 \right) (n+1) + c^{2n+1} \left(\frac{\sigma_1}{\sigma_2} n + n + 1 \right) \left(\frac{\sigma_2}{\sigma_3} n + n + 1 \right) \right] \\ & \cdot \left[\left(\frac{\sigma_3}{\sigma_4} n + n + 1 \right) + (n+1) \left(\frac{\sigma_3}{\sigma_4} - 1 \right) d^{2n+1} \right] \\ & + (n+1) c^{2n+1} \left[b^{2n+1} \left(\frac{\sigma_1}{\sigma_2} - 1 \right) \left(\frac{\sigma_2}{\sigma_3} n + \frac{\sigma_2}{\sigma_3} + n \right) + c^{2n+1} \left(\frac{\sigma_1}{\sigma_2} n + n + 1 \right) \left(\frac{\sigma_2}{\sigma_3} - 1 \right) \right] \\ & \cdot \left[n \left(\frac{\sigma_3}{\sigma_4} - 1 \right) + \left(\frac{\sigma_3}{\sigma_4} n + \frac{\sigma_3}{\sigma_4} + n \right) d^{2n+1} \right] \end{aligned} \quad (\text{A.11})$$

where

- b , outer-most radius (relative to the radius of the sphere R) for the brain

$$b = r_1 / R$$
- c , outer-most radius (relative to the radius of the sphere R) for the cerebrospinal fluid

$$c = r_2 / R$$
- d , outer-most radius (relative to the radius of the sphere R) for the skull

$$d = r_3 / R$$
- $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are the conductivity of the brain, cerebrospinal fluid, skull and scalp respectively.

Appendix B

Approximation

In this appendix it is explained roughly how to eliminate the infinite summation in the forward equation (1). In order to understand deeply all the approximation process it is suggested to read the paper [23] written by Mingui Sun.

► Key Concepts

Let us remind the forward equation (A.9):

$$v = \frac{1}{4\pi\sigma_4 R^2} \sum_{n=1}^{\infty} c_n f^{n-1} m \cdot \left[r_0 P_n(\cos \theta) + t_0 \frac{P_n^1(\cos \theta)}{n} \right]$$

The next figure (figure B.1) represents the c_n parameter (A.10) as a function of n for the four-shell model of Cuffin and Cohen [20].

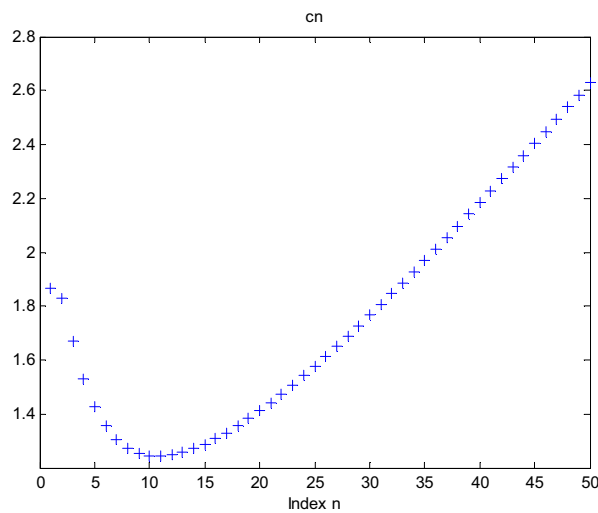


Figure B.1. Values of c_n with respect to n .

As it is showed, except to the first few values of c_n (on the left), the remaining c_n values tend to follow a smooth v -shaped curve which asymptotically approaches a straight line. This observation induces to evaluate the equation (1) by separating the infinite sum into two parts. The first part involves the first few (two or three) terms of (1) using the exact values of c_n . On the other hand, the second part is composed of the remaining values which will be fitting with a polynomial.

The advance in this second part is due to only the beginning portion of this fitting has to be accurate because of the fact that as n increases, f^{n-1} exponentially decreases (figure B.2). Thus, it will be neglected higher-order terms. Detailed discussions about this approach are provided in [23].

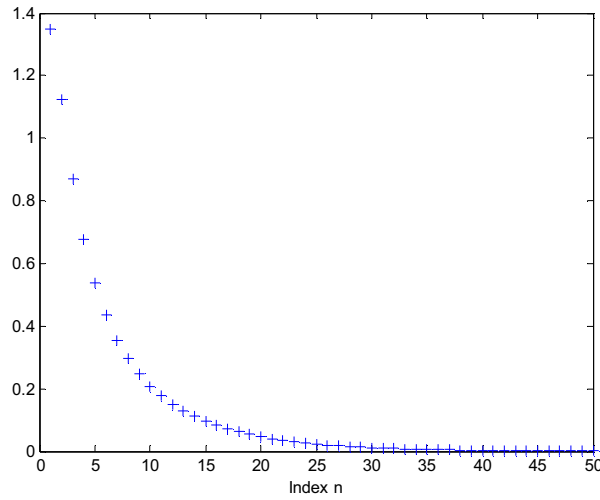


Figure B.2. Values of $c_n f^{n-1}$ with respect to n and $f = 0.85$.

► Method

As it was explained, the N lower-order terms of c_n are separated from the remaining terms. These higher-order terms are fitted with a polynomial of order K

$$v \approx \bar{v} = \sum_{n=1}^N c_n f^{n-1} Q_n + \sum_{n=N+1}^{\infty} \left(\sum_{k=0}^K a_k n^k \right) f^{n-1} Q_n \quad (\text{B.1})$$

where

$$Q_n = \frac{1}{4\pi\sigma_4 R^2} m \cdot \left[r_0 P_n(\cos\theta) + t_0 \frac{P_n^1(\cos\theta)}{n} \right] \quad (\text{B.2})$$

In order to evaluate (B.1) Mingui Sun determines the polynomial coefficients a_k and simplifies the infinite sums to closed-form expression. In his work, he also concludes that N and K should be chosen as small as possible in order to maximize the efficiency.

The final equation is showed in B.3 (all the new parameters in this formula can be checked in [23]).

$$v \approx \bar{v} = \frac{m}{4\pi\sigma_4 R^2} \cdot \left\{ \sum_{n=1}^N \tilde{c}_n f^{n-1} \left[r_0 P_n(x) + \frac{t_0 P_n^1(x)}{n} \right] + \sum_{k=0}^K a_k (r_0 L_k + t_0 M_k) \right\} \quad N = 3, K = 3 \quad (\text{B.3})$$

As it is showed in equation B.3, the initial infinite sums have been reduced into two sums of finite terms.

Appendix C

BESA

This appendix provides information about MEGIS software and introduces the reader to the clinical application used in this thesis (DipoleSimulator).

C.1 MEGIS Software GmbH

This software was created by Dr. Michael Scherg in 1995. Products of MEGIS Software GmbH are the leading innovator in digital EEG and MEG software for research and clinical applications in the field of:

- Electroencephalography (EEG)
- Magnetoencephalography (MEG)
- Evoked Potentials (EP) and Fields (EF)
- Event-Related Potentials (ERP) and Fields (ERF)
- Combined Neuroimaging (EEG, MEG, MRI, fMRI)

The goal of this software is to offer advanced technologies for analysis and visualization of human brain activity. This goal is possible thanks to experts in different fields of neuroscience and with the latest computational techniques. Results are BESA and FOCUS programs, whose capabilities are appreciated by users in clinical facilities and research institutes.

C.1.1 What does BESA provide?

BESA (Brain Electrical Source Analysis) is a program for source analysis and dipole localization in EEG and MEG research. Although source analysis is the core of the BESA program, BESA provides excellent tools for the pre-processing of your raw data.

BESA also provides a variety of source analysis algorithms, a standardized realistic head model (FEM), and allows integration with MRI (magnetic resonance imaging) and fMRI (functional magnetic resonance imaging).

C.1.2 What does EEGFocus provide?

EEGFocus offers a review and advanced EEG analysis. The development of computerized EEG has brought enormous advantages over the traditional paper EEG. EEGFocus can transform the scalp EEG into signals of source activity which predominantly reflect the electrical activity of specific regions in the brain. This program uses advanced analysis features such as 3D head mapping, brain source montages, spike pattern search and spectral analysis. EEGFocus provides a user-friendly interface for immediate analysis of abnormal patterns during review, e.g. spikes and seizure data of epilepsy patients.

C.1.3 DipoleSimulator

DipoleSimulator is a free program written by Patrick Berg for the simulation of EEG or MEG activity generated by model sources. The purpose of this tool is to provide a better understanding of dipole modelling.

C.2 DipoleSimulator

C.2.1 Program Version

The program used is a free download version:

Dipole Simulator Version 3.1,0,6, Aug 18 2006 Copyright © Patrick Berg, 2001-2006
--

Several parts of the program sources were taken from the sources of the BESA2000 program with compliments of MEGIS software. Program updates can be downloaded from the MEGIS website <http://www.besa.de>

► References

- Whole-head schemes are Copyright © Dr. Michael Scherg, 2000.
- Maps use spherical splines as published by Perrin et al. (1988) and Pascual-Marqui et al. (1988).
- Pascual-Marqui, R.D., Gonzalez-Andino, S.L., Valdez-Sosa, P.A., Biscay-Lirio, R. “Current source density estimation and interpolation based on the spherical harmonic fourier expansion”. *Intern. J. Neurosci.*, 43, 237-249 (1998).

- Perrin, F., Pernier, J., Bertrand, O., Echallier, J.F. "Spherical splines for scalp potential and current source density mapping". *Electroenceph. Clin. Neurophysiol.*, 72, 184-187 (1988).

C.2.2 Introduction

DipoleSimulator is a program for generating and visualizing data simulations based on spatio-temporal dipole models. With the program you can perform the following operations:

- Simulate both EEG and MEG.
- See maps resulting from a dipole anywhere within the head.
- Add up to 20 dipoles to a model.
- Generate independent waveforms of each dipole (source waveform). You can also read in source waveforms from an ASCII file (BESA swf).
- See the surface data resulting from the model.
- Place a cursor anywhere in the time interval and view maps of the data at that time point.
- Specify the parameters of the spherical head model.
- Add coherent noise to the data.
- Learn about filters by displaying the effects of applying various types of filter to your simulated data.

In the next sections first of all it will be described the DipoleSimulator window and the most important settings that can be tuned. The next step will be to analyse the different divisions of the DipoleSimulator window. And finally it will be explained some concepts about the simulated noise.

C.2.3 The DipoleSimulator Window

In the next figure (figure C.1) the DipoleSimulator window is showed.

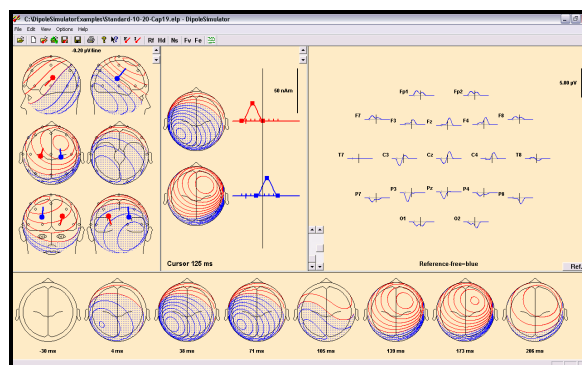


Figure C.1. DipoleSimulator Window

C.2.4 Settings

▶ Head Model

Edit > Head Model

When this item is selected, the following dialog box appears:

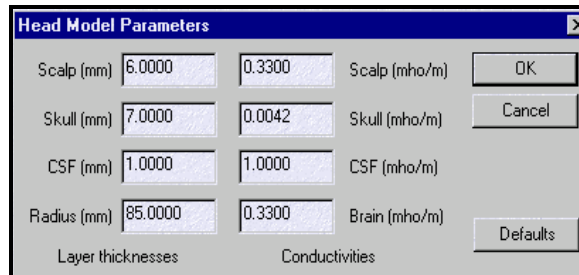


Figure C.2. Head Model Parameters Box

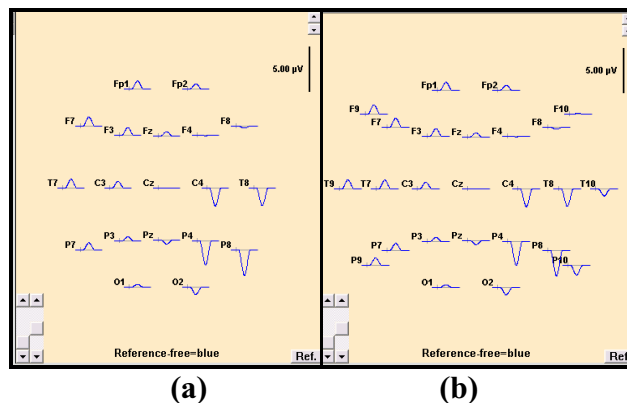
EEG data are generated using a spherical four-shell head model (Berg and Scherg, 1994). The default parameters are shown in the previous box. Typing in the dialog box you can generate different head models.

▶ EEG Sensors

File > Load EEG/MEG sensors

Here you can select the sensor system model just reading in an electrode file or MEG sensor files. Initially you can choose between the Standard 10-20 (19 and 25 electrodes) or the Standard 10-10 (33 and 81 electrodes), figure C.3.

The format of the electrode file is that used by the BESA program (*.elp). Each electrode requires one line of an ASCII file, containing 2 parameters, *theta* and *phi* (in degrees). *Theta* records the angle down from the (vertical) axis linking the centre of the head with Cz. *Phi* records the angle in the horizontal plane with the line linking the two ears.



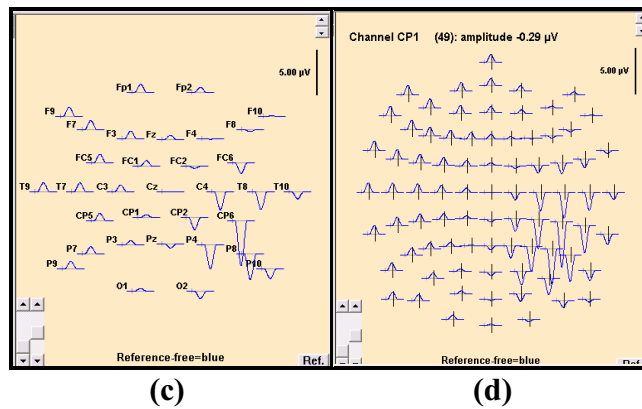


Figure C.3. Sensor Systems. Waveforms are shown in a topographic view with the nose at the top. If there are more than 50 channels, labels will not be displayed **(a)** Standard 10-20, 19 electrodes. **(b)** Standard 10-20, 25 electrodes. **(c)** Standard 10-10, 33 electrodes. **(d)** Standard 10-10, 81 electrodes.

► **Reference**

Edit > Reference Channel

When this item is selected, the following dialog box appears:

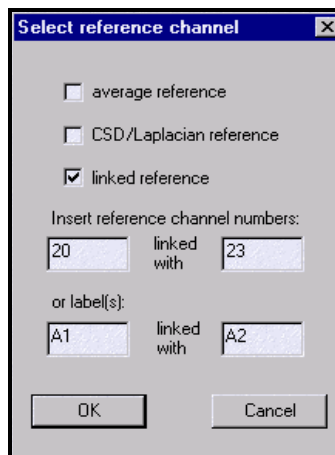


Figure C.4. Reference Box

The potential of each electrode can be expressed by different methods. The default data waveforms are reference free, i.e they are calculated directly from the dipole field. In the first option each channel is defined by an **average reference**, which means that the potential of each electrode is compared to the average of all electrodes. Reference free and average referenced data are often almost identical when you include electrodes below the midline. On the other hand it is less similar when the electrodes cover only the upper half of the head.

Furthermore, you can select CSD (**Laplacian**) reference by clicking in the middle checkbox or compare the potential of each electrode to a neutral electrode just clicking in the third (**linked reference**) checkbox before defining the channels.

C.2.5 Heads Frame

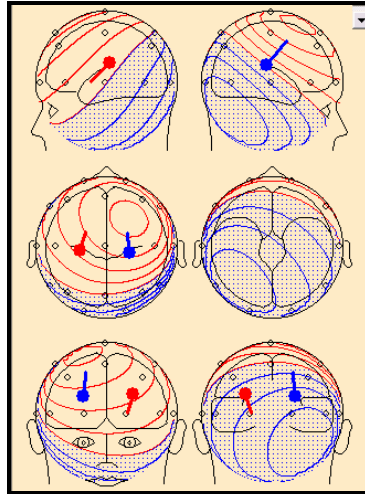


Figure C.5. Heads frame. Example with two dipoles within the head model.

The six heads are for editing dipole location and orientation and for displaying data maps. You can add or delete sources with a mouse double click and change the source location and orientation with click and drag. You can also edit each source (*Edit > Edit Sources*) using the dialog box:

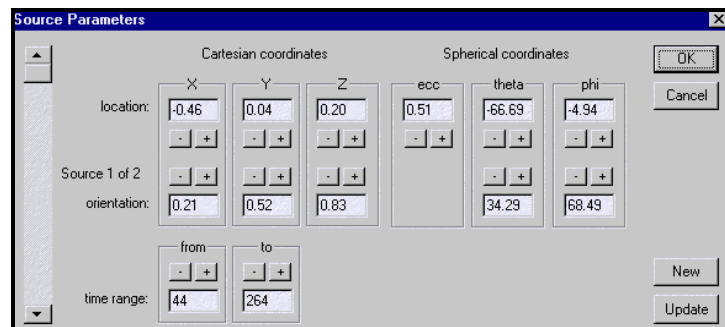


Figure C.6. Source Parameters Box

Select the source to edit with the scrollbar on the left of the dialog box. Insert values for the location and orientation, and for the time range of the source waveform. In this process when you change values, other values are not immediately updated. When you exit with **OK** or when you press the **Update** button, coordinates are checked for consistency so that the source remains within the unit sphere and the orientation vector has unit amplitude. Then the values are changed in the display.

C.2.6 Dipole Maps

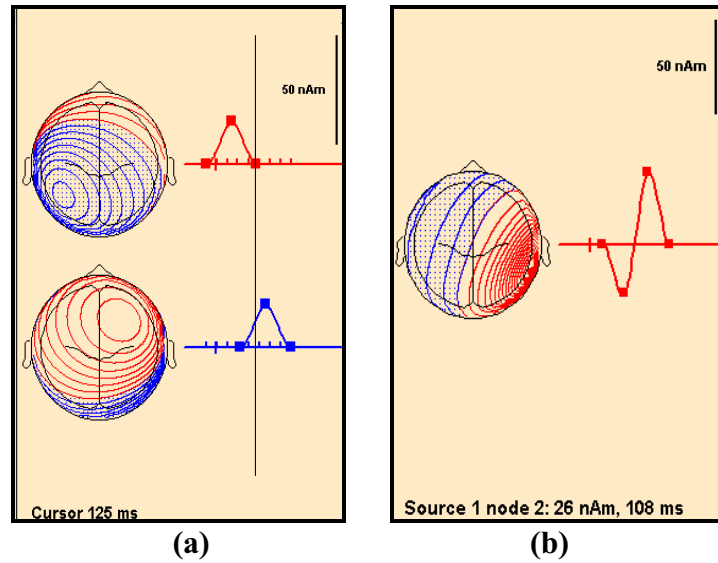


Figure C.7. Dipole maps (a) Example with two dipoles within the head model (b) Example with four nodes.

This map (figure C.7.a) shows the topography of single sources using a fixed arbitrary scale, and the input source waveforms.

You can edit source waveforms here. Nodes can be dragged to a new latency (end nodes) or new amplitude (intermediate nodes). Furthermore, new nodes can be created or deleted with a double click on the horizontal line (figure C.7.b). In order to display the amplitude and latency of the source waveform you have to switch on the cursor with a double click over it and move the cursor by dragging it with the mouse.

In figure C.7, at the right top corner, there is a line where you can adjust the source waveform scaling just clicking over it.

C.2.7 Output Data Waveform Display

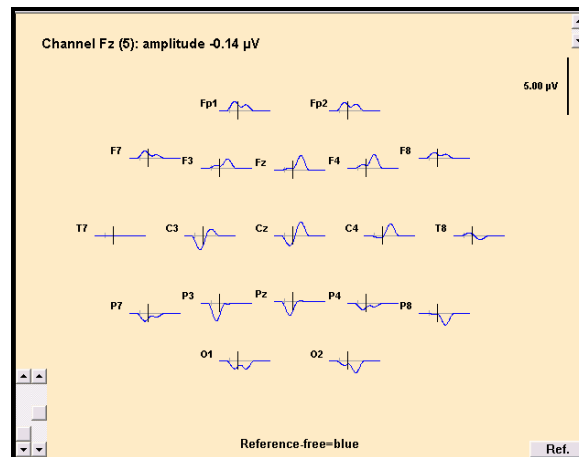


Figure C.8. Output Waveform display using Standard 10-20 (19 sensors).

As it was said previously, waveforms are shown in a topographic view with the nose at the top. To get the voltage for a certain channel, switch the cursor on and move the mouse over the channel.

Use the line at the right top corner in order to adjust data waveform amplitudes and the **Ref** button to select the reference method.

The left scrollbar adjusts the amplitude of coherent noise which is added to the simulated data. The right scrollbar adjusts the relative amplitude of 10 Hz noise (simulated alpha) in the noise spectrum.

C.2.8 Map Area

Edit > Map Sequence

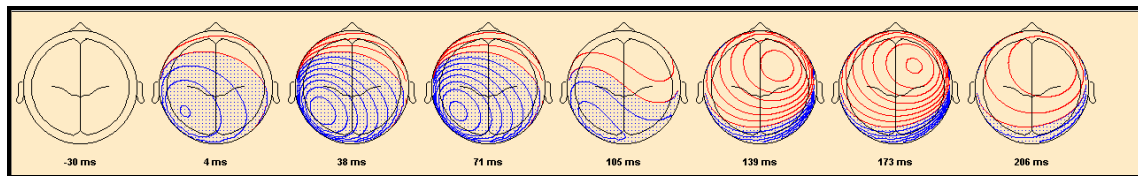


Figure C.9. Map area.

In figure C.9 it is drawn a sequence of map latencies. Parameters of the sequence like viewpoint, range and interval can be set in the *Edit > Map Sequence* dialog box. Maps are generated using spherical splines (Perrin et al.)

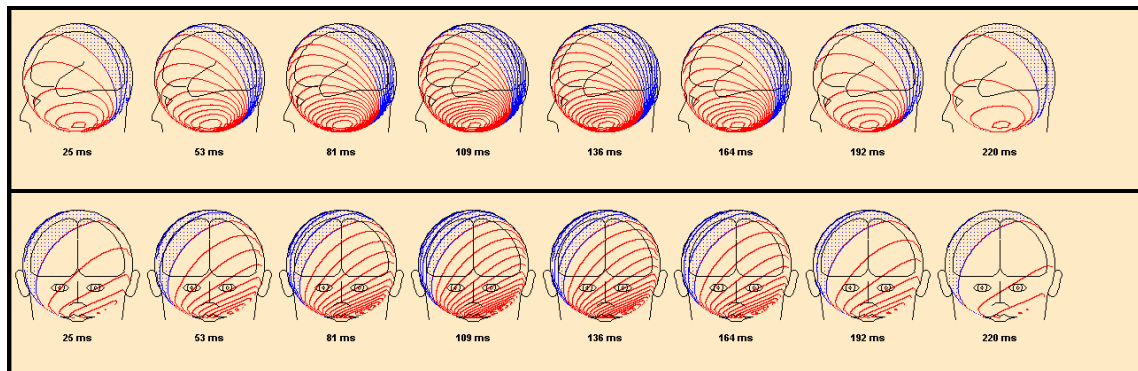


Figure C.10. Map areas. Example of left and front viewpoints.

C.2.9 How noise is defined

The aim of the simulated noise is to try to give it some of the properties of background EEG. For example, the noise is coherent in the sense that there is quite a high correlation between signal amplitudes from electrodes that are close together.

Noise is defined using the following main steps:

- The waveform generated in the program is assumed to consist of 150 time points, sampled at 100 Hz, representing an interval of 1.5 s.
- 200 random sources are generated subject to the following restrictions:
 1. Eccentricity lies between 0.6 and 0.7 of the head radius
 2. Location is not below 0.5 of the head radius below the sphere centre.
 3. Orientation is random.
- For each source, its waveform is defined in frequency space.
- For each source, the data waveform at each electrode is generated.
- For each electrode, data waveforms are summed over the contributions from all 200 sources.
- Over all waveforms and electrodes, the average referenced data are scaled to unit standard deviation.
- These noise data are then added in the user-specified proportion to the dipole simulated data.

Appendix D

Source Codes

In this appendix the main source codes are provided with detailed comments. In the first part D.1 and D.2 the *10-20 Electrode System* and the *Potential Response* are developed in **Matlab**. In the second part the Particle Swarm Optimization implementation is presented. It consists of an overview about the PSO code structure and the PSO source codes developed in C.

D.1 10-20 Electrode Source Code

```
%-----  
% Sphere.m  
% Head Model: Sphere (R=10 cm)  
%           Electrodes according to the 10-20 standar system  
%           19 sensores+ 2 references  
%-----  
hold on  
sphere % draw a sphere with radio 1 dm.  
circle=rsmak('circle',1,[0,0]);  
fnplt(circle);  
  
%From Polar to Cartesian coordinate-----  
%Data values calculated with perimeter 60 cm  
  
[x,y,z]=sph2cart(0,(360-24.91)/180*pi,1);%Naison  
[x20,y20,z20]=sph2cart(0,(180+24.91)/180*pi,1);%Inion  
[x21,y21,z21]=sph2cart(pi/2,(180+24.91)/180*pi,1);%A1 (reference)  
[x22,y22,z22]=sph2cart(pi/2,(360-24.91)/180*pi,1);%A2 (reference)  
[x23,y23,z23]=sph2cart(pi/2,pi/2,1)%C0
```

Appendix D

```
[x2,y2,z2]=sph2cart(0,44.04/180*pi,1)%F0
[x3,y3,z3]=sph2cart(0,135.96/180*pi,1)%P0
[x4,y4,z4]=sph2cart(18/180*pi,-1.93/180*pi,1);%FP1
[x5,y5,z5]=sph2cart(54/180*pi,-1.93/180*pi,1);%F7
[x6,y6,z6]=sph2cart(pi/2,-1.93/180*pi,1);%T3
[x7,y7,z7]=sph2cart(-54/180*pi,(180+1.93)/180*pi,1);%T5
[x8,y8,z8]=sph2cart(-18/180*pi,(180+1.93)/180*pi,1);%O1
[x9,y9,z9]=sph2cart(-18/180*pi,-1.93/180*pi,1);%FP2
[x10,y10,z10]=sph2cart(-54/180*pi,-1.93/180*pi,1);%F8
[x11,y11,z11]=sph2cart(pi/2,181.93/180*pi,1);%T4
[x12,y12,z12]=sph2cart(54/180*pi,(180+1.93)/180*pi,1);%T6
[x13,y13,z13]=sph2cart(18/180*pi,(180+1.93)/180*pi,1);%O2
[x14,y14,z14]=sph2cart(pi/2,44.04/180*pi,1);%C4
[x15,y15,z15]=sph2cart(pi/2,135.96/180*pi,1);%C3
[x16,y16,z16]=sph2cart(36/180*pi,(33.88)/180*pi,1);%F3
[x17,y17,z17]=sph2cart(-36/180*pi,(33.88)/180*pi,1);%F4
[x18,y18,z18]=sph2cart(36/180*pi,(180-33.88)/180*pi,1);%P4
[x19,y19,z19]=sph2cart(-36/180*pi,(180-33.88)/180*pi,1);%P3
```

```
%-----
plot3(x,y,z,'ro');
plot3(x2,y2,z2,'r*');
plot3(x22,y22,z22,'r*');
plot3(x23,y23,z23,'r*');
plot3(x3,y3,z3,'r*');
plot3(x4,y4,z4,'r*');
plot3(x5,y5,z5,'r*');
plot3(x10,y10,z10,'r*');
plot3(x12,y12,z12,'r*');
plot3(x9,y9,z9,'r*');
plot3(x17,y17,z17,'r*');
plot3(x15,y15,z15,'r*');
plot3(x11,y11,z11,'r*');
plot3(x13,y13,z13,'r*');
plot3(x8,y8,z8,'r*');
plot3(x16,y16,z16,'r*');
plot3(x14,y14,z14,'r*');
plot3(x6,y6,z6,'r*');
plot3(x7,y7,z7,'r*');
plot3(x18,y18,z18,'r*');
plot3(x19,y19,z19,'r*');
plot3(x20,y20,z20,'r*');
plot3(x21,y21,z21,'r*');
hold off
```

D.2 Potential Equation Source Code

```

%-----
% Esf_efi.m
% Four layers head model voltage response.
% Used approximation: "An efficient algorithm for computing multishell
% spherical volume conductor models in EEG dipole source localization"
% Mingui Sun, Member IEEE [10]
%-----

close all
clear all
format long

%Sensors position %-----
[x_n,y_n,z_n]=sph2cart(0,(360-24.91)/180*pi,1);%Naison
[x_i,y_i,z_i]=sph2cart(0,(180+24.91)/180*pi,1);%Inion
%A1(reference
[x_ref1,y_ref1,z_ref1]=sph2cart(pi/2,(180+24.91)/180*pi,1);
%A2(reference)
[x_ref2,y_ref2,z_ref2]=sph2cart(pi/2,(360-24.91)/180*pi,1);
[x1,y1,z1]=sph2cart(pi/2,pi/2,1)%C0
[x2,y2,z2]=sph2cart(0,44.04/180*pi,1)%F0
[x3,y3,z3]=sph2cart(0,135.96/180*pi,1);%P0
[x4,y4,z4]=sph2cart(18/180*pi,-1.93/180*pi,1);%FP1
[x5,y5,z5]=sph2cart(54/180*pi,-1.93/180*pi,1);%F7
[x6,y6,z6]=sph2cart(pi/2,-1.93/180*pi,1);%T3
[x7,y7,z7]=sph2cart(-54/180*pi,(180+1.93)/180*pi,1);%T5
[x8,y8,z8]=sph2cart(-18/180*pi,(180+1.93)/180*pi,1);%O1
[x9,y9,z9]=sph2cart(-18/180*pi,-1.93/180*pi,1);%FP2
[x10,y10,z10]=sph2cart(-54/180*pi,-1.93/180*pi,1);%F8
[x11,y11,z11]=sph2cart(pi/2,181.93/180*pi,1);%T4
[x12,y12,z12]=sph2cart(54/180*pi,(180+1.93)/180*pi,1);%T6
[x13,y13,z13]=sph2cart(18/180*pi,(180+1.93)/180*pi,1);%O2
[x14,y14,z14]=sph2cart(pi/2,44.04/180*pi,1);%C4
[x15,y15,z15]=sph2cart(pi/2,135.96/180*pi,1);%C3
[x16,y16,z16]=sph2cart(36/180*pi,(31.88)/180*pi,1);%F3
[x17,y17,z17]=sph2cart(-36/180*pi,(31.88)/180*pi,1);%F4
[x18,y18,z18]=sph2cart(36/180*pi,(180-31.88)/180*pi,1);%P4
[x19,y19,z19]=sph2cart(-36/180*pi,(180-31.88)/180*pi,1)%P3

sensors=[x1 y1 z1;x2 y2 z2;x3 y3 z3;x4 y4 z4;x5 y5 z5;x6 y6 z6;x7 y7 z7;x8
y8 z8;x9 y9 z9;x10 y10 z10;x11 y11 z11;x12 y12 z12;x13 y13 z13;x14 y14
z14;x15 y15 z15;x16 y16 z16;x17 y17 z17;x18 y18 z18;x19 y19 z19];

```

Appendix D

```

%Input -----
r=[-0.77 0.42 -0.13];% r=(rx,ry,rz) dipole location vector
m=[-0.10*(300e-3) 0.52*(300e-3) 0.85*(300e-3)];% m=(mx,my,mz) current
dipole moment (mA.cm)

% Shpere model-----
R1=8.5; %Brain radio(cm)
R2=8.7; %CFS(cerebrospinal fluid) radio(cm)
R3=9.4; %Skull radio(cm)
R4=10; %Scalp radio(cm)
r=r.*R4;
sigma1=33; %Brain conductivity (mho.cm)
sigma2=100; %CFS conductivity (mho.cm)
sigma3=0.42; %Skull conductivity (mho.cm)
sigma4=33; %Scalp conductivity (mho.cm)

% N and K should be chosen as small as possible according to (27)-----
N=3;
K=3;

% Evaluating cn (equation (2) and (3))-----
n=(1:300);
x=(2*n+1);

F=(...
(R3/R4).^x.*...
(((R1/R4).^x).^n.*((sigma1/sigma2)-1).*((sigma2/sigma3)-1).*(n+1))+
((R2/R4).^x).^((sigma1/sigma2).*n+n+1).*((sigma2/sigma3).^n+n+1)).*
(((sigma3/sigma4).^n+n+1)+(n+1).*((sigma3/sigma4)-1).*((R3/R4).^x) )...
)+...
(...
((n+1).*((R2/R4).^x)).*...
(((R1/R4).^x).^((sigma1/sigma2)-1).*((sigma2/sigma3).^n+
(sigma2/sigma3)+n))+((R2/R4).^x).^((sigma1/sigma2).*n+n+1).*
((sigma2/sigma3)-1))).*...
((n.*((sigma3/sigma4)-1))+((sigma3/sigma4).^n+(sigma3/sigma4)+n).*
((R3/R4).^x) ));

cn=((x.^4).*( ( (R2/R4)*(R3/R4) ).^x))./F;

plot(n(1:50),cn(1:50),'+')
title('cn');
xlabel('Index n');
figure
f(1,:)=0.85.^(n(2:51));

```

Appendix D

```

cn_f=cn(1:50).*f;
plot(n(1:50),cn_f,'+');
xlabel('Index n');

% Evaluating ak (equation (30)). Case K=3-----
mm=(N+1:300);% infinitive
ohm=(mm.*(2.*mm-1))./((R1/R4).^2.*mm);
cnn=cn(mm);

b0=sum(cnn./ohm);
b1=sum((cnn.*mm)./ohm);
b2=sum((cnn.*(mm.^2))./ohm);
b3=sum((cnn.*(mm.^3))./ohm);
b=[b0;b1;b2;b3]; % vector b
a0=sum(1./ohm);
a1=sum(mm./ohm);
a2=sum(mm.^2./ohm);
a3=sum(mm.^3./ohm);
a4=sum(mm.^4./ohm);
a5=sum(mm.^5./ohm);
a6=sum(mm.^6./ohm);
A=[a0 a1 a2 a3;a1 a2 a3 a4; a2 a3 a4 a5; a3 a4 a5 a6]; %Matrix A

a=inv(A)*b; % Ax=b system of ecuations (30);
% x=a=[a0;a1;a2;a3]= ak coefficients
a=a'; %vector

%Evaluating cn' (equation 6) -----
p=(1:N);
indi=(0:K);

cnPrima_0=0;
cnPrima_1=cn(1)-sum(a);
cnPrima_2=cn(2)-sum(a.*((2).^indi));
cnPrima_3=cn(3)-sum(a.*((3).^indi));
cnPrima=[cnPrima_1 cnPrima_2 cnPrima_3];

% Calculate Potencial V (mv)-----

for i=1:19 %number of sensors

sx=sensors(i,1);%s=(sx,sy,sz) surface point (cm)
sy=sensors(i,2);
sz=sensors(i,3);
mult2=R4/(sqrt(sx^2+sy^2+sz^2));% mod s = R4

```

```

sx=mult2*sx;
sy=mult2*sy;
sz=mult2*sz;
s=[sx sy sz]; %vector sensor

C=1/(4*pi*sigma4);
R2=sum(s.*s); % R^2=sx^2+sy^2+sz^2
R=sqrt(R2); % R=sqrt(sx^2+sy^2+sz^2)=mod_s
mod_r=sqrt(sum(r.*r)); % |r|
f=mod_r/R;

%Legendre polynomials-----
q=sum(r.*s);
x=q/(mod_r*R); %x=cos(teta), teta=angle between r and s
P1=x;
P2=0.5*(3*x^2-1);
P3=0.5*(5*x^3-3*x);

%Associated Legendre polynomials-----
P11=sqrt(1-x^2);
P12=3*x*P11;
P13=(3/2)*(5*(x^2)-1)*P11;

%Lk coefficients-----
aux=sqrt(1-2*x*f+f^2);
L0=((1/aux)-1)/f;
L1=(x-f)*(1/aux^3);
L2=(x+(x^2)*f-x*(f^2)-2*f+f^3)*(1/aux^5);
L3=(x-4*f+5*(x^2)*f-9*x*(f^2)+10*(f^3)+(f^2)*(x^3)-2*(x^2)*(f^3)-x*(f^4)-f^5)*(1/aux^7);

% Mk coefficients-----
M0=(1+aux)/(aux*(1-f*x+aux))*P11;
M1=(1/aux^3)*P11;
M2=(1/aux^5)*(1+f*x-2*f^2)*P11;
M3=(1/aux^7)*(1-10*(f^2)+4*(f^2)+5*f*x-(f^3)*x+(f^2)*x^2)*P11;

T=s*sum(r.*r)-r*sum(r.*s);
aux=sum(T.*T);
mod_t=sqrt(aux);
if (aux==0)
    to=0;
else
    to=sum(m.*T)/mod_t; % t0 vector, include m.to
end

```

Appendix D

```
ro=sum(m.*r)/mod_r; % r0 vector, include m.ro
if (f==0) %case A: central dipole, when f=0
V(i)=cn(1)*C*(sum(m.*s))/(R2*R); % ecuation (31)
else %case B= radial dipole
vr=(cnPrima(1)*P1)+(cnPrima(2)*f*P2)+(cnPrima(3)*f^2*P3)+(a(1)*L0)+(a(2)*L1
)+(a(3)*L2)+(a(4)*L3);
vf=(cnPrima(1)*P11)+(cnPrima(2)*P12*f/2)+(cnPrima(3)*P13*(f^2)/3)+(a(1)*M0)
+(a(2)*M1)+(a(3)*M2)+(a(4)*M3);
V(i)=C*(to*vf+ro*vr)/R2;
end;
end

figure;
hold on
plot(0,'*');
plot(V,'o');
hold off
```

D.3 Particle Swarm Optimization Implementation

D.3.1 Code Structure

The original PSO source codes are provided on the web site [-24] as a supplement for the book “Swarm Intelligence” [24]. This code implements basic swarm algorithm to optimize five benchmark functions. Major modifications of the original PSO codes are: (a) The Absorbing Wall boundary condition for constraints on particle movement; (b) The optimization problem.

The PSO source codes can be divided into three parts:

PSO.C implements the swarm algorithm.

MYFUN.C implements the 6D optimization problem (sphere_model function).

PSO.RUN specifies the settings for the PSO.

An example of PSO.RUN file is as follows:

```
10.0 0.0 0.1 0.1 0.1 0 10 -1 1 -1 1 100 Srun Cost Best 0.9 5 6 5
```

```
//Example Setting File PSO.RUN
```

```
NUMBER_OF_AGENTS
```

```
SCALEMULT
```

```
E_CUTOFF
```

```
MAXV_1
```

```
MAXV_2
```

```
MAXX
```

```
IRang_L, IRang_R
```

```
Rad_Rang_L, Rad_Rang_R
```

```
Moment_Rang_L, Moment_Rang_R
```

```
MAXITER
```

```
results_file
```

```
Cost_file
```

```
Best_file
```

```
weight
```

```
fun_type: 0-Shaffer f6
```

```
1-Sphere
```

```
2-Rosenbrock
```

```
3-generalized Rastrigrin
```

```
4-generalized Griewank
```

```
5-Potencial Min Cost
```

```
DIMENSION
```

```
run_no
```


The above file specifies the next PSO settings: **10** particles; maximum location and orientation velocity **0.1**; dynamic range of the search space for location (r_x, r_y, r_z) and orientation (m_x, m_y, m_z) parameters within **[-1, 1]**; maximum iteration **100**; the prefix for the resulting text files **Srun, Cost** and **Best**; the inertial weight **0.9**; the optimization problem developed in this thesis **5**; the dimension of the optimization problem ($r_x, r_y, r_z, m_x, m_y, m_z$) **6**; total number of separate runs **5**.

D.3.2 How to Run the PSO Code

After compiled `PSO.c`, `PSO.exe` is created. To run the file, simply type `PSO PSO.RUN` at the DOS prompt. When the maximum iteration is reached, the program will be concluded and `Srun.txt`, `Cost.txt` and `Best.txt` text files will be produced.

► Text files description

`SrunX.txt` : each line represents a 6D position ($r_x, r_y, r_z, m_x, m_y, m_z$) for one particle at certain iteration.

`CostX.txt` : each line represents the global best cost at certain iteration.

`BestX.txt` : each line represents the global best position at certain iteration.

(X represents the number of PSO runs).

D.3.3 PSO Structure Sketch

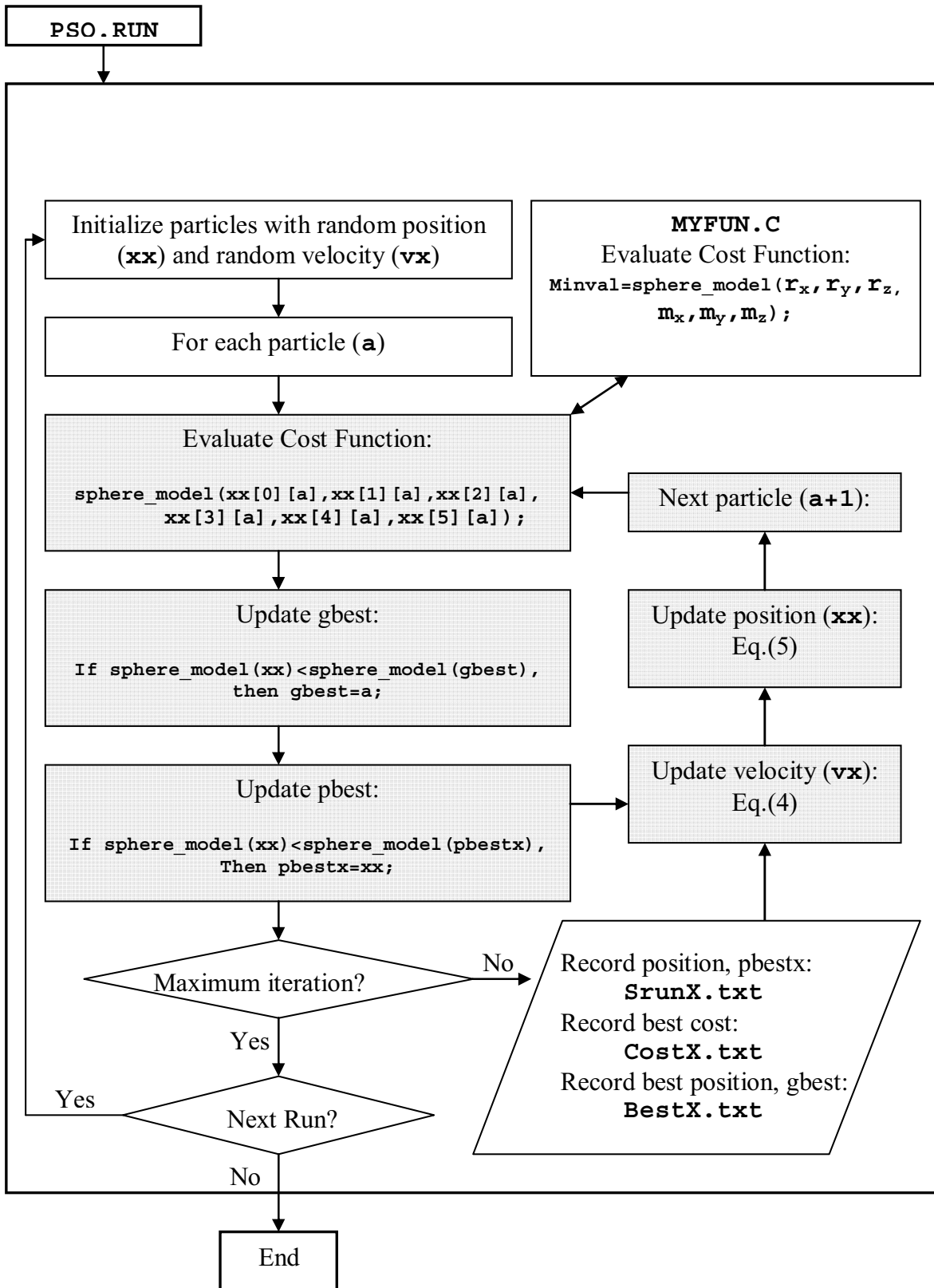


Figure D.1. PSO Structure Sketch

D.3.4 PSO Source Code

PSO.C

```
/*
The original program optimizes five benchmark functions using swarm
algorithm asynchronous version by Yuhui Shi, May 15, 1998.
Modified by Laura Prada with a new optimization problem: sphere_model
function.
*/

#include "headfile.h"
#include "global.h"
#include "mem_loc.h"
#include <math.h>
#include "myfun.h"
#include "randomlib.h"
#include "sphcarte.h"

/* ***** main() ***** */
main (int argc, char *argv[])
{
    int NUMBER_OF_AGENTS;
    int MAXITER;
    float E_CUTOFF, MAXV_1,MAXV_2, MAXX, SCALEMULT;
    float weight, weight_up;
    int run_no; //nombre of runs

    FILE *fp, *frun;
    char runfile[60], resfile[60];
    char temp[10];
    char tempfile[60];
    double minval=0.0;
    int DIMENSION;
    int fun_type;
    //0:Schaffer f6, 1:sphere, 2:Rosenbrock, 3:generalized Rastrigrin
    //4:generalized Griewank, 5:Sphere_model

    float IRang_L, IRang_R; // initialization rang: left and right
                           range
    float Rad_Rang_L,Rad_Rang_R;
    float Moment_Rang_L,Moment_Rang_R;
    float error,error_num,error_den,real_rx,real_ry,real_rz,real_mx,
        real_my,real_mz;
    int a,b;
```

Appendix D

```
int i;
int iter;
int gbest;
int firsttime;
int tmp,finish;
float aux0,aux1,aux2;

time_t tt;

/* *****
   Open runfile
   ***** */
if (argc<2)
{
    printf("Need to specify runfile");
    exit(1);
}
strcpy(runfile,argv[1]);

if ((frun=fopen(runfile,"r"))==NULL)
{
    printf("Cant read file");
    exit(1);
}

/* N G S E V X M U Res */

fscanf(frun, "%d %f %f %f %f %f %f %f %f %f %f %d %s %f %d %d
%d",&NUMBER_OF_AGENTS,
        &SCALEMULT,
        &E_CUTOFF,&MAXV_1,&MAXV_2,
        &MAXX,&IRang_L,&IRang_R,&Rad_Rang_L,&Rad_Rang_R,&Moment_Rang_L,&Moment_Rang
_R, &MAXITER, tempfile,&weight, &fun_type,&DIMENSION,&run_no);
fclose(frun);

FVectorAllocate(&pbest,
                NUMBER_OF_AGENTS);
FVectorAllocate(&maxx,
                DIMENSION);
FMatrixAllocate(&vx,
                DIMENSION,NUMBER_OF_AGENTS);
FMatrixAllocate(&xx,
                DIMENSION,NUMBER_OF_AGENTS);
FMatrixAllocate(&tx,
                DIMENSION,NUMBER_OF_AGENTS);
FMatrixAllocate(&pbestx,
                DIMENSION,NUMBER_OF_AGENTS);

for (a=0;a<DIMENSION;a++)
{
    maxx[a]=MAXX;
    /* range of xx[] */
}
```

Appendix D

```
time(&tt);
printf("begin time: %s\n",ctime(&tt));

//loop for runs
for (i=0;i<run_no;i++)
{
    firsttime=1; //first iteration of this run
    iter=0;
    gbest=0; //initialy assume the first particle as the gbest

/* *****
    This loop initializes the individual agents for each run
    ***** */

for (a=0;a<NUMBER_OF_AGENTS;a++)
{
    if (fun_type==5)
    {

xx[0][a] =Rad_Rang_R*RandomUniform(); //initialize Radial Position rx
if (RandomUniform() > 0.5) xx[0][a]=-xx[0][a];
xx[1][a] =Rad_Rang_R*RandomUniform(); //initialize Radial Position ry
if (RandomUniform() > 0.5) xx[1][a]=-xx[1][a];
xx[2][a]=Rad_Rang_R *RandomUniform();//initialize Radial Position rz
if (RandomUniform() > 0.5) xx[2][a]=-xx[2][a];
xx[3][a]=Moment_Rang_R*RandomUniform();//initialize dipole moment mx
if (RandomUniform() > 0.5) xx[3][a]=-xx[3][a];
xx[4][a]=Moment_Rang_R*RandomUniform();//initialize dipole moment my
if (RandomUniform() > 0.5) xx[4][a]=-xx[4][a];
xx[5][a]=Moment_Rang_R*RandomUniform();//initialize dipole moment mz
if (RandomUniform() > 0.5) xx[5][a]=-xx[5][a];

for (b=0;b<2;b++) // b<DIMENSION
{
    pbestx[b][a]=xx[b][a];
    if (b==0)
    {
        vx[b][a] = Rad_Rang_R*RandomUniform();
        if (RandomUniform() > 0.5) vx[b][a]=-vx[b][a];
        vx[b+1][a]=Rad_Rang_R*RandomUniform();
        if (RandomUniform() > 0.5) vx[b+1][a]=-vx[b+1][a];
        vx[b+2][a]=Rad_Rang_R*RandomUniform();
        if (RandomUniform() > 0.5) vx[b+2][a]=-vx[b+2][a];
    }
}
```

```

if (b==1)
{
vx[b+2][a] = Moment_Rang_R*RandomUniform();
if (RandomUniform() > 0.5) vx[b+2][a]=-vx[b+2][a];
vx[b+3][a]=Moment_Rang_R*RandomUniform();
if (RandomUniform() > 0.5) vx[b+3][a]=-vx[b+3][a];
vx[b+4][a]=Moment_Rang_R*RandomUniform();
if (RandomUniform() > 0.5) vx[b+4][a]=-vx[b+4][a];
}
} //end for
} //end if (fun_type==5)

else //for benchmark functions
{
for (b=0;b<DIMENSION;b++)
{
xx[b][a] = (float) ((IRang_R - IRang_L)*(rand()/32767.0) +
IRang_L);
pbestx[b][a]=xx[b][a];
vx[b][a] = MAXV_1*(rand()/32767.0);
if ((rand()/32767.0) > 0.5) vx[b][a]=-vx[b][a];
} //end for
} //end else
} //end for

/* *****
Main Work Loop for each run here
***** */

/*****
Get the output result file name
*****/

if (i>=10)
{
int temdec=i/10;
temdec=temdec+48;
strcpy(temp, (char*)&temdec);
tmp=i%10 +48;
strcat(temp, (char*)&tmp);
}
else
{
int tmp=i+48;

```

```
strcpy(temp, (char*)&tmp);
}
strcpy(resfile, tempfile);
strcat(resfile, temp);
strcat(resfile, ".txt");

/*****
    Open file for output best agent index vs iteration
*****/

if ((fp=fopen(resfile, "w"))==NULL)
{
printf("Cant write file");
exit(1);
}

finish=0;
fprintf(fp, "Random initialization as follows:\n");

for (a=0;a<NUMBER_OF_AGENTS;a++)
{

/* *****/
    Record the initial coordinates
*****/
for (b=0;b<DIMENSION;b++)
{
fprintf(fp, "%f ", xx[b][a]); //record the position
}
fprintf(fp, "\n");
}
fprintf(fp, "\n");
do
{
iter++;
if (iter >32760) iter=0; /* so it doesnt crash the data type */

//update inertia weight
//weight_up = (weight-0.4) * (MAXITER - iter) /MAXITER +0.4;
//time variant weight, linear from weight to 0.4

weight_up=weight;          //constant inertia weight

for (a=0;a<NUMBER_OF_AGENTS;a++)
```

```

{
/* *****
   Cost Function Evaluations
   Error is returned by function routines
***** */
switch (fun_type)
{
  case 0:
    minval=f6(a);
    break;
  case 1:
    minval=sphere(a,DIMENSION);
    break;
  case 2:
    minval=rosenbrock(a,DIMENSION);
    break;
  case 3:
    minval=rastrigrin(a,DIMENSION);
    break;
  case 4:
    minval=griewank(a,DIMENSION);
    break;
  case 5:
    minval=sphere_model(xx[0][a],xx[1][a],xx[2][a],
                       xx[3][a],xx[4][a],xx[5][a]);

    fprintf(fp," min val= %f",minval);
    break;
  default:
    printf("\n Not a valid function type\n");
    exit(1);
}

if (firsttime==1) pbest[a]=minval;
if (minval < pbest[a]) //update pbest
{
pbest[a]=minval;
for (b=0;b<DIMENSION;b++) pbestx[b][a]=xx[b][a];
if (pbest[a] < pbest[gbest]) gbest=a; //update gbest
}

/* asynchronous version */
for (b=0;b<DIMENSION;b++)
{

```



```

//velocity update
vx[b] [a] = weight_up*vx[b] [a] + 1.49*(RandomUniform())*(pbestx[b] [a] -
xx[b] [a]) +1.49*(RandomUniform())*(pbestx[b] [gbest] -xx[b] [a]);

//velocity limit
if ((b==0) || (b==1) || (b==2))
{
if (vx[b] [a]>MAXV_1)
vx[b] [a]=MAXV_1;
else if (vx[b] [a]<-MAXV_1)
vx[b] [a]=-MAXV_1;
}
if ((b==3) || (b==4) || (b==5))
{
if (vx[b] [a]>MAXV_2)
vx[b] [a]=MAXV_2;
else if (vx[b] [a]<-MAXV_2)
vx[b] [a]=-MAXV_2;
}
}

/*****
Tx allows simultaneous updates
*****/
for (b=0; b<DIMENSION;b++)
{
tx[b] [a]=xx[b] [a]+vx[b] [a];

if (fun_type==5)
{
//to check if the particle hit the boundary
if ((b==0) || (b==1) || (b==2))
{
if (tx[b] [a] < Rad_Rang_L || tx[b] [a] > Rad_Rang_R)
{
vx[b] [a]=0; //absorbing wall
tx[b] [a]=xx[b] [a]+vx[b] [a];
}
} //end inner if
if ((b==3) || (b==4) || (b==5))
{
if (tx[b] [a]< Moment_Rang_L || tx[b] [a]> Moment_Rang_R)
{
vx[b] [a]=0; //absorbing wall
tx[b] [a]=xx[b] [a]+vx[b] [a];
}
}
}
}

```

```

    }
    }//end inner if
} //end outer if(fun_type==5)
} //end for
} / END OF a LOOP

/* *****
   Update positions
***** */

for (a=0;a<NUMBER_OF_AGENTS;a++)
{
/* *****
   Define new coordinates
***** */
for (b=0;b<DIMENSION;b++)
{
xx[b][a] =tx[b][a];
fprintf(fp,"%f  ",xx[b][a]);
}
fprintf(fp, "\n");
} /* end a loop */

/* *****
   In case iterations become greater than 32767
***** */
if (firsttime!=1)
{
if (iter==32766) iter=0;
}
/* *****
   Terminate on criterion
***** */
//output best index vs iteration
if (fun_type==0)
{
fprintf(fp,"%f\n",1.0-pbest[gbest]);
}
else
{
fprintf(fp,"Global Min Cost:  %f\n\ Position: ",pbest[gbest]);
for (b=0;b<DIMENSION;b++)
fprintf(fp, "%f ", pbestx[b][gbest]);
fprintf(fp, "\n\n");
}
}

```

```

if ((pbest [gbest] <= E_CUTOFF) || (iter >= MAXITER))
{
fclose(fp);
printf("%d run finished!\n",i);
finish=1;
}
firsttime=0;

/* ***** End of do-loop ***** */
while (! finish);
}
/*****
    Root mean square error
*****/

real_rx= -0.77;
real_ry= 0.42; //change values(X-Y)
real_rz= -0.13;
real_mx= -0.1;
real_my= 0.52;
real_mz= 0.85;

// Error Orientation+Position
error_num1=pow((real_rx-pbestx[0] [gbest]),2)+pow((real_ry-
pbestx[1] [gbest]),2) + pow((real_rz-pbestx[2] [gbest]),2)+ pow((real_mx-
pbestx[3] [gbest]),2)+pow((real_my-pbestx[4] [gbest]),2) +pow((real_mz-
pbestx[5] [gbest]),2);
error1=sqrt(error_num1)*100;

//Error Position
error_num2=pow((real_rx-pbestx[0] [gbest]),2)+pow((real_ry-
pbestx[1] [gbest]),2) + pow((real_rz-pbestx[2] [gbest]),2);
error2=sqrt(error_num2)*100;

//Error Orientation
error_num3=pow((real_mx-pbestx[3] [gbest]),2)+pow((real_my-
pbestx[4] [gbest]),2) + pow((real_mz-pbestx[5] [gbest]),2);
error3=sqrt(error_num3)*100;

printf("error_orienta+position (%) = %f\n",error1);
printf("error_position (%) = %f\n",error2);
printf("error_orientation (%)= %f \n", error3);

/*****

```

```
time(&tt);
printf("end time: %s\n",ctime(&tt));

//release the memory allocated for PSO
free(pbest);
free(maxx);
FMatrixFree(vx, DIMENSION);
FMatrixFree(xx, DIMENSION);
FMatrixFree(tx, DIMENSION);
FMatrixFree(pbestx, DIMENSION);
return 0;
}
```

MYFUN.C

```

/*
The 6D cost function: sphere_model.
Written by Laura Prada.
*/

#include "headfile.h"
#include "extern.h"
#include "myfun.h"
#include "sphcarte.h"
#define pi 3.14159264
#include <math.h>

double sphere_model(double rx,double ry,double rz,double mx, double
my,double mz)
{

/*****
Calculate the potencial for a dipole with positon r (rx ry rz) [cm]
and dipole moment m (mx my mz) [mA.cm]
*****/

float sensors[19][3];

//Sensors' position. Satandar system 10-20: perimeter 28.5 cm

sensors[0][0]=sph2cart_x(90,90,1);//C0
sensors[0][1]=sph2cart_y(90,90,1);
sensors[0][2]=sph2cart_z(90,90,1);

sensors[1][0]=sph2cart_x(0,44.04,1);//F0
sensors[1][1]=sph2cart_y(0,44.04,1);
sensors[1][2]=sph2cart_z(0,44.04,1);

sensors[2][0]=sph2cart_x(0,135.96,1);//P0
sensors[2][1]=sph2cart_y(0,135.96,1);
sensors[2][2]=sph2cart_z(0,135.96,1);

sensors[3][0]=sph2cart_x(18,-1.93,1);//FP1
sensors[3][1]=sph2cart_y(18,-1.93,1);
sensors[3][2]=sph2cart_z(18,-1.93,1);

sensors[4][0]=sph2cart_x(54,-1.93,1);//F7
sensors[4][1]=sph2cart_y(54,-1.93,1);

```

Appendix D

```
sensors[4][2]=sph2cart_z(54,-1.93,1);

sensors[5][0]=sph2cart_x(90,-1.93,1);//T3
sensors[5][1]=sph2cart_y(90,-1.93,1);
sensors[5][2]=sph2cart_z(90,-1.93,1);

sensors[6][0]=sph2cart_x(-54,(180+1.93),1);//T5
sensors[6][1]=sph2cart_y(-54,(180+1.93),1);
sensors[6][2]=sph2cart_z(-54,(180+1.93),1);

sensors[7][0]=sph2cart_x(-18,(180+1.93),1);//O1
sensors[7][1]=sph2cart_y(-18,(180+1.93),1);
sensors[7][2]=sph2cart_z(-18,(180+1.93),1);

sensors[8][0]=sph2cart_x(-18,-1.93,1);//FP2
sensors[8][1]=sph2cart_y(-18,-1.93,1);
sensors[8][2]=sph2cart_z(-18,-1.93,1);

sensors[9][0]=sph2cart_x(-54,-1.93,1);//F8
sensors[9][1]=sph2cart_y(-54,-1.93,1);
sensors[9][2]=sph2cart_z(-54,-1.93,1);

sensors[10][0]=sph2cart_x(90,181.93,1);//T4
sensors[10][1]=sph2cart_y(90,181.93,1);
sensors[10][2]=sph2cart_z(90,181.93,1);

sensors[11][0]=sph2cart_x(54,(180+1.93),1);//T6
sensors[11][1]=sph2cart_y(54,(180+1.93),1);
sensors[11][2]=sph2cart_z(54,(180+1.93),1);

sensors[12][0]=sph2cart_x(18,(180+1.93),1);//O2
sensors[12][1]=sph2cart_y(18,(180+1.93),1);
sensors[12][2]=sph2cart_z(18,(180+1.93),1);

sensors[13][0]=sph2cart_x(90,44.04,1);//C4
sensors[13][1]=sph2cart_y(90,44.04,1);
sensors[13][2]=sph2cart_z(90,44.04,1);

sensors[14][0]=sph2cart_x(90,135.96,1);//C3
sensors[14][1]=sph2cart_y(90,135.96,1);
sensors[14][2]=sph2cart_z(90,135.96,1);

sensors[15][0]=sph2cart_x(36,(31.88),1);//F3
sensors[15][1]=sph2cart_y(36,(31.88),1);
```

Appendix D

```
sensors[15][2]=sph2cart_z(36,(31.88),1);

sensors[16][0]=sph2cart_x(-36,(31.88),1);//F4
sensors[16][1]=sph2cart_y(-36,(31.88),1);
sensors[16][2]=sph2cart_z(-36,(31.88),1);

sensors[17][0]=sph2cart_x(36,(180-31.88),1);//P4
sensors[17][1]=sph2cart_y(36,(180-31.88),1);
sensors[17][2]=sph2cart_z(36,(180-31.88),1);

sensors[18][0]=sph2cart_x(-36,(180-31.88),1);//P3
sensors[18][1]=sph2cart_y(-36,(180-31.88),1);
sensors[18][2]=sph2cart_z(-36,(180-31.88),1);

//Head model-----

float R1=8.5; // Brain radio(cm)
float R2=8.7; // CFS(cerebrospinal fluid) radio(cm)
float R3=9.4; // Skull radio(cm)
float R4=10; // Scalp radio(cm)
float r[3],m[3];

r[0]=rx*R4;
r[1]=ry*R4;
r[2]=rz*R4;
m[0]=mx*30e-3;
m[1]=my*30e-3;
m[2]=mz*30e-3;

float sigma1=33; // Brain conductivity (mho.cm)
float sigma2=100; // CFS conductivity (mho.cm)
float sigma3=0.42; // Skull conductivity (mho.cm)
float sigma4=33; // Scalp conductivity (mho.cm)

//N and K should be chosen as small as possible according to (27)
int N=3;
int K=3;

//Data-----
int p,xx,yy,i,k,j,n,mm,x,y,nn,x1,y1,w,q;
float s[3],T[3],cnPrima[3],V_Sensors[19];
float result,mult2,C,R22,R,mod_r,f,P1,P2,P3,P11,P12,P13,aux,L0,L1,L2,
L3,M0,M1,M2,M3,Y,mod_t,to,ro,fact1,fact2,fact3,qq,x_poli,vr,vf;
float F[300],cn[300],ohm[300],cnn[300];
float b[4],A[4][8],ak[4],Inv_A[4][4],r_vec[3],rs_vec[3];
```

Appendix D

```

// Evaluating cn (equation (2) and (3))-----
for (n=0;n<300;n++)
{
nn=n+1;
x=(2*nn+1);

F[n]=( pow((R3/R4),x)*( ((pow((R1/R4),x))*nn*( (sigma1/sigma2)-1 )*(
(sigma2/sigma3)-1 )*(nn+1)) + ( pow((R2/R4),x))*( (sigma1/sigma2)*nn+nn+1
)*( (sigma2/sigma3)*nn+nn+1)))*( ((sigma3/sigma4)*nn+nn+1)+(nn+1)*(
(sigma3/sigma4)-1 )*(pow((R3/R4),x)) ))+( ( nn+1)*(pow((R2/R4),x)) )*(
((pow((R1/R4),x))*( (sigma1/sigma2)-1 )*(
(sigma2/sigma3)*nn+(sigma2/sigma3)+nn))+ ( pow((R2/R4),x))*(
(sigma1/sigma2)*nn+nn+1)*((sigma2/sigma3)-1)))*( nn*((sigma3/sigma4)-1)) +
( ((sigma3/sigma4)*nn+(sigma3/sigma4)+nn)*(pow((R3/R4),x)) ) );
cn[n]=((pow(x,4))*( pow(( (R2/R4)*(R3/R4) ),x)))/F[n];
}

// Evaluating ak (equation (30)). Case K=3-----
for(mm=(N+1);mm<301;mm++) // infinitiv (300 itera.)
{
ohm[mm-N-1]=(mm*(2*mm-1))/(pow((R1/R4),(2*mm)));
cnn[mm-N-1]=cn[mm-1];
}

b[0]=0;
b[1]=0;
b[2]=0;
b[3]=0;
A[0][0]=0;
A[0][1]=0;
A[0][2]=0;
A[0][3]=0;
A[1][3]=0;
A[2][3]=0;
A[3][3]=0;

mm=N;
for (i=0;i<299-N;i++)
{
mm=mm+1;

b[0]=b[0]+(cnn[i]/ohm[i]); // vector b
b[1]=b[1]+((cnn[i]*mm)/ohm[i]);
b[2]=b[2]+((cnn[i]*(pow(mm,2)))/ohm[i]);
b[3]=b[3]+((cnn[i]*(pow(mm,3)))/ohm[i]);
}

```



```

A[0][0]=A[0][0]+(1/ohm[i]);
A[0][1]=A[0][1]+(mm/ohm[i]);
A[0][2]=A[0][2]+(pow(mm,2)/ohm[i]);
A[0][3]=A[0][3]+(pow(mm,3)/ohm[i]);
A[1][3]=A[1][3]+(pow(mm,4)/ohm[i]);
A[2][3]=A[2][3]+(pow(mm,5)/ohm[i]);
A[3][3]=A[3][3]+(pow(mm,6)/ohm[i]);
}
A[1][0]=A[0][1];
A[1][1]=A[0][2];
A[1][2]=A[0][3];
A[2][0]=A[0][2];
A[2][1]=A[0][3];
A[2][2]=A[1][3];
A[3][0]=A[0][3];
A[3][1]=A[1][3];
A[3][2]=A[2][3];

// Inverse of A: Inv_A-----
n=4; //dimension matrix
for (int xx=0;xx< n;xx++)
{
for (int yy=0;yy< n;yy++)
{
if (yy==xx)
A[xx][yy+n]= 1;
else
A[xx][yy+n]= 0;
}
}
float c ;
int cont,sepudo;
for (x=0;x<n;x++)
{
cont = 0;
sepudo = 1;
do//
{
cont++;
c=A[x][x];
if (c == 0)
if (cont+x <= n)
{
for (int yyy =0;yyy<=n;yyy++)//cambiar(x,x+cont);
{

```

```

float aux = A[x][yyy];
A[x][yyy] = A[x+cont][yyy];
A[x+cont][yyy] = aux;
} //for 2
c = A[x][x];
} //if 1
else
sepudo = 0;
}
while( (c == 0) && sepudo);
if (sepudo)
{
for (y =x;y<2*n;y++)
A[x][y] = A[x][y] / c;
for( x1 = x+1;x1<n;x1++)
{
c = A[x1][x];
for (y1 = x;y1<2*n;y1++)
A[x1][y1] = A[x1][y1] - c * A[x][y1];
}
for (x1 = x-1;x1>=0;x1--)
{
c = A[x1][x];
for (y1 = x;y1<2*n;y1++)
A[x1][y1] = A[x1][y1] - c * A[x][y1];
}
}
} // end for 1
for (q=0;q<4;q++)
for (w=4;w<8;w++)
    Inv_A[q][w-4]=A[q][w];

// Ax=b system of equations (30)-----
//x=a=[a0;a1;a2;a3]= ak coefficients

for(i=0;i<4;i++)
{
for(j=0;j<1;j++)
{
ak[i]=0;
for(k=0;k<4;k++)
    ak[i]+=Inv_A[i][k]*b[k];
}
}
}

```

Appendix D

```
//Evaluating cn' (equation (6)) -----
fact1=0;
fact2=0;
fact3=0;
for (p=0;p<4;p++)
{
fact1=fact1+ak[p];
fact2=fact2+ak[p]*pow(2,p);
fact3=fact3+ak[p]*pow(3,p);
}
cnPrima[0]=cn[0]-fact1;
cnPrima[1]=cn[1]-fact2;
cnPrima[2]=cn[2]-fact3;

// Calculate Potencial V (v) ( equation (19)) -----

for (i=0;i<19;i++) //number of sensors ...19
{ //ini for sensor
mult2=R4/(sqrt(
pow(sensors[i][0],2)+pow(sensors[i][1],2)+pow(sensors[i][2],2) ));// mod s
= R4
s[0]=mult2*sensors[i][0];
s[1]=mult2*sensors[i][1];
s[2]=mult2*sensors[i][2];
C=1/(4*pi*sigma4);
R22=pow(s[0],2)+pow(s[1],2)+pow(s[2],2); // R^2=sx^2+sy^2+sz^2
R=sqrt(R22); //R=sqrt(sx^2+sy^2+sz^2)=mod_s
mod_r=sqrt(pow(r[0],2)+pow(r[1],2)+pow(r[2],2));// |r|
f=mod_r/R;

//Legendre polynomials-----
qq=r[0]*s[0]+r[1]*s[1]+r[2]*s[2];
x_poli=qq/(mod_r*R); //x=cos(teta), teta=angle between r and s
P1=x_poli;
P2=0.5*(3*pow(x_poli,2)-1);
P3=0.5*(5*pow(x_poli,3)-3*x_poli);

//Associated Legendre polynomials-----
P11=sqrt(1-pow(x_poli,2));
P12=3*x_poli*P11;
P13=1.5*(5*pow(x_poli,2)-1)*P11;

//Lk coefficients-----
aux=sqrt(1-2*x_poli*f+pow(f,2));
L0=((1/aux)-1)/f;
```

Appendix D

```

L1=(x_poli-f)*(1/(pow(aux,3)));
L2=(x_poli+(pow(x_poli,2))*f - x_poli*(pow(f,2))- 2*f
+pow(f,3))*(1/pow(aux,5));
L3=(x_poli- 4*f + 5*(pow(x_poli,2))*f - 9*x_poli*(pow(f,2)) +
10*(pow(f,3)+ (pow(f,2))*(pow(x_poli,3)) - 2*(pow(x_poli,2))*(pow(f,3)) -
x_poli*(pow(f,4)) - pow(f,5))*(1/pow(aux,7));

// Mk coefficients-----
M0=(1+aux)/(aux*(1-f*x_poli+aux))*P11;
M1=(1/pow(aux,3))*P11;
M2=(1/pow(aux,5))*(1+f*x_poli-2*pow(f,2))*P11;
M3=(1/pow(aux,7))*(1- 10*(pow(f,2))+ 4*(pow(f,2))+ 5*f*x_poli -
(pow(f,3))*x_poli + (pow(f,2))*pow(x_poli,2))*P11;

r_vec[0]=(r[0]*r[0]+r[1]*r[1]+r[2]*r[2])*s[0];
r_vec[1]=(r[0]*r[0]+r[1]*r[1]+r[2]*r[2])*s[1];
r_vec[2]=(r[0]*r[0]+r[1]*r[1]+r[2]*r[2])*s[2];
rs_vec[0]=(r[0]*s[0]+r[1]*s[1]+r[2]*s[2])*r[0];
rs_vec[1]=(r[0]*s[0]+r[1]*s[1]+r[2]*s[2])*r[1];
rs_vec[2]=(r[0]*s[0]+r[1]*s[1]+r[2]*s[2])*r[2];

T[0]=r_vec[0] -rs_vec[0]; //equation Appendix (pag 704)
T[1]=r_vec[1] -rs_vec[1];
T[2]=r_vec[2] -rs_vec[2];
aux=T[0]*T[0]+T[1]*T[1]+T[2]*T[2];
mod_t=sqrt(aux);

if (aux==0)
to=0;
else
to=(m[0]*T[0]+m[1]*T[1]+m[2]*T[2])/mod_t; // t0 vector, include m.to
ro=(m[0]*r[0]+m[1]*r[1]+m[2]*r[2])/mod_r; // r0 vector, include m.ro

if (f==0) //case A: central dipole, when f=0

// equation (31)
V_Sensors[i]=cn[0]*C*100*(m[0]*s[0]+m[1]*s[1]+m[2]*s[2]);
else //case B= radial dipole
{
vr=(cnPrima[0]*P1)+(cnPrima[1]*f*P2)+(cnPrima[2]*pow(f,2)*P3)+(ak[0]*L0)+(a
k[1]*L1)+(ak[2]*L2)+(ak[3]*L3);
vf=(cnPrima[0]*P11)+(cnPrima[1]*P12*f/2)+(cnPrima[2]*P13*(pow(f,2))/3)+(ak[
0]*M0)+(ak[1]*M1)+(ak[2]*M2)+(ak[3]*M3);
V_Sensors[i]=C*1000*((to*vf)+(ro*vr)); // V_sensors*1e-5
V_Sensors[i]=C*1000*((to*vf)+(ro*vr))/R22;
}

```

Appendix D

```
}
} //en for sensor

/*****
Original Position - Potential Measured
Dipole simulator values (reference simulator!)*1e-5
*****/

//Case 1: r(0 0 0), m(0 0.63 0.78), 31nA.m, ecc 0
//double V_Simulator[19]= {1.16*1e-1,1.47*1e-1, 0.14*1e-1, 0.84*1e-1
,0.51*1e-1 ,-0.04*1e-1, -0.59*1e-1, -0.92*1e-1, 0.84*1e-1, 0.51*1e-1, -
0.04*1e-1, -0.59*1e-1, -0.92*1e-1, 0.80*1e-1, 0.80*1e-1, 1.20*1e-1,
1.20*1e-1, -0.05*1e-1, -0.05*1e-1};

//Case 2: r(0 0.5 0.3), m(0 0.63 0.78), 31nA.m, ecc 0.58
//double V_Simulator[19]={1.05*1e-1, 2.93*1e-1, -0.14*1e-1 ,0.82*1e-1
,0.17*1e-1 ,-0.33*1e-1 ,-0.60*1e-1, -0.71*1e-1, 0.82*1e-1, 0.17*1e-1, -
0.33*1e-1, -0.60*1e-1, -0.71*1e-1, 0.51*1e-1, 0.51*1e-1, 1.62*1e-1,
1.62*1e-1 ,-0.27*1e-1, -0.27*1e-1};

//Case 3: r(-0.25 0.20 -0.25), m(0 -0.71 0.71), 30nA.m, ecc 0.4
//double V_Simulator[19]= {0.90*1e-1, 0.16*1e-1, 1.14*1e-1, -0.91*1e-1, -
0.39*1e-1, 0.11*1e-1, 0.53*1e-1, 0.83*1e-1, -1.16*1e-1, -0.67*1e-1,
0.27*1e-1, 0.84*1e-1, 0.97*1e-1, 0.60*1e-1, 0.88*1e-1, 0.01*1e-1, 0.04*1e-
1, 1.17*1e-1 ,0.90*1e-1};

// Case 4: r(-0.25 0.24 0.42), m(0 -0.89 -0.45), 30nA.m, ecc 0.53
//double V_Simulator[19]={-0.65*1e-1, -2.27*1e-1, 0.79*1e-1, -1.04*1e-1, -
0.43*1e-1, 0.18*1e-1, 0.66*1e-1, 1.01*1e-1, -1.30*1e-1, -0.69*1e-1,
0.37*1e-1, 1.01*1e-1, 1.16*1e-1, -0.21*1e-1, -0.29*1e-1, -1.18*1e-1, -
2.15*1e-1, 0.96*1e-1, 0.66*1e-1};

//Case 5: r(0 0.5 0.6), m(0 0.69 0.72),30 nA.m, ecc 0.72
//double V_Simulator[19]= {0.99*1e-1, 4.72*1e-1, -0.40*1e-1, 0.59*1e-1,
0.07*1e-1, -0.39*1e-1, -0.65*1e-1, -0.76*1e-1, 0.59*1e-1, 0.07*1e-1, -
0.39*1e-1, -0.65*1e-1, -0.76*1e-1, 0.28*1e-1, 0.28*1e-1, 1.47*1e-1,
1.47*1e-1, -0.48*1e-1 ,-0.48*1e-1};

//Case 6: r(0 -0.59 0.52), m(0 0.61 0.79),30 nA.m, ecc 0.78
//double V_Simulator[19]={1.97*1e-1 ,1.08*1e-1, 0.94*1e-1 ,0.40*1e-1
,0.27*1e-1 ,-0.06*1e-1 ,-0.73*1e-1 ,-1.72*1e-1 ,0.40*1e-1 ,0.27*1e-1 ,-
0.06*1e-1 ,-0.73*1e-1 ,-1.72*1e-1, 0.99*1e-1 ,0.99*1e-1 ,0.82*1e-1, 0.82
*1e-1,-0.16*1e-1, -0.16*1e-1 };
```

Appendix D

```
//Case 7: r(0 -0.7 0.36), m(0.57 0.52 0.64),30 nA.m, ecc 0.8
//double V_Simulator[19]={1.31*1e-1 ,0.82*1e-1, 1.18*1e-1 ,0.49*1e-1
,0.66*1e-1 ,0.70*1e-1 ,0.39*1e-1 , -1.03*1e-1 ,0.24*1e-1 , -0.11*1e-1 , -
0.63*1e-1 , -1.45*1e-1, -2.40*1e-1 ,1.44*1e-1 ,0.15*1e-1, 0.94*1e-1
,0.37*1e-1, -1.02*1e-1, 1.37*1e-1};

//Case 8: r(-0.05 -0.07 0.54), m(0.68 -0.68 -0.26),30 nA.m, ecc 0.54
//double V_Simulator[19]= {-0.73*1e-1, -1.27*1e-1, 1*1e-1, -0.44*1e-1
,0.25*1e-1 ,0.89*1e-1, 1.29*1e-1, 1.24*1e-1, -0.96*1e-1, -1.13*1e-1, -
0.84*1e-1, -0.12*1e-1, 0.70*1e-1, 0.80*1e-1, -1.55*1e-1, -0.27*1e-1, -
1.65*1e-1, 0.04*1e-1, 1.49*1e-1};

//Case 9:r(0.45 0.50 0.56), m(0.58 -0.73 -0.34),30 nA.m, ecc 0.87
//double V_Simulator[19]={-0.51*1e-1 , -2.79*1e-1, 0.48*1e-1, -0.99*1e-1,
0.63*1e-1, 1.44*1e-1, 1.25*1e-1, 0.85*1e-1, -1.22*1e-1, -0.82*1e-1, -
0.37*1e-1, 0.05*1e-1, 0.45*1e-1, 1.38*1e-1, -0.66*1e-1 , -1.90*1e-1 , -
1.52*1e-1, 0.10*1e-1, 1.07*1e-1};

//Case 10:r(0.42 -0.77 -0.13), m(0.52 -0.10 0.85),30 nA.m, ecc 0.89
//double V_Simulator[19]={0.79*1e-1, 0.32*1e-1 ,1.18*1e-1, -0.02*1e-1,
0.27*1e-1, 0.78*1e-1, 1.99*1e-1, 1.14*1e-1, -0.22*1e-1, -0.37*1e-1, -
0.5*1e-1 , -0.59*1e-1, -0.52*1e-1, 1.26*1e-1, 0.12*1e-1, 0.48*1e-1, 0.01*1e-
1, 0.19*1e-1, 2.37*1e-1};

//Case 11:r(0.55 0.52 0.57), m(0.55 -0.68 -0.49),30 nA.m, ecc 0.94
double V_Simulator[19]= {-0.79*1e-1, -2.70*1e-1, 0.31*1e-1, -0.83*1e-1,
0.85*1e-1, 1.58*1e-1, 1.24*1e-1 ,0.80*1e-1, -1.02*1e-1, -0.68*1e-1, -
0.31*1e-1, 0.05*1e-1, 0.41*1e-1, 1.10*1e-1, -0.68*1e-1, -3.51*1e-1, -
1.38*1e-1, 0.02*1e-1, 0.92*1e-1};

/*****
Cost Function
*****/
result=0;
for (i=0;i<19;i++)
{
result=result+(100*V_Sensors[i]-100*V_Simulator[i])*(100*V_Sensors[i]-
100*V_Simulator[i]); // result*1e-14
}
printf("%f \n",result);
return result;
```