Bayesian Approach to the Ill-posed EEG Inverse Problem

Thorsteinn Már Arinbjarnarson

Kongens Lyngby 2007
IMM-2007-77

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

Summary

Scalp recorded EEG signals are caused by neural currents in the brain. The brain currents are believed to be related to behavior and cognition. Estimating the current density from EEG recordings is the inverse EEG problem. The EEG inverse problem is highly underdetermined and some assumptions have to be made when solving it. Here assumptions will be made in the form of probability distributions describing the neural current distribution and the signal noise. Bayes theorem enables detailed analytical calculations to be made. These calculations lead to equations used to formulate iterative algorithms. Basic Gaussian distribution assumption gives a simple and robust algorithm. Automatic relevance determination (ARD) is used to from a more sparse current estimate. An original update formula is presented in the ARD algorithm which the author has not found elsewhere. Smoothing is also incorporated into the algorithms to account for localized currents.

Simulations are presented for evaluation purposes of the different methods. These tests show the different properties of the algorithms. The Gaussian algorithm converges fast and is the most robust. For sparse sources the ARD algorithm gives better estimates but for realistic forward models it usually fails. Adding spatial smoothing into the ARD iteration improves its performance and good results are obtained for realistic forward models. An attempt is made to incorporate smoothing into the Bayesian framework but the resulting algorithm does not perform better than the ARD one. Finally some real data is analyzed using the algorithms.

Preface

I think and think for months and years. Ninety-nine times, the conclusion is false. The hundredth time I am right.
A. Einstein

After struggling a bit with getting some of my theory into code I was reading through a booklet on creativity by *Victor Vidal* [33], a professor at IMM (Informatics and Mathematical Modelling), my department here at the Technical University of Denmark. In it I came across the Einstein quotation above, it immediately lifted my spirit. This was in the early stages of my project which started in January 2007 and along the way I have surely made countless wrong conclusions and written a whole lot of useless code. But now seven months later close to the deadline in August my work has lead to some correct conclusions and efficient code which forms the basis of this MSc thesis. *Lars Kai Hansen* professor at the Intelligent Signal Processing Group has given me solid guidance along the way, having kept me on track while giving me freedom to challenge myself.

Kgs. Lyngby, August 2007

Þorsteinn Már Arinbjarnarson [s053120]

Acknowledgements

I would like to thank my supervisor Professor Lars Kai Hansen for his guidance. Morten Mørup I thank for mathematical discussions, on topics such as linear algebra and regularization of ill-posed problems. Tom Bolwig and Troels Kjær from Rigshospitalet have encouraged EEG brain imaging research at the department and provided interesting discussion on their work with EEG and their vision on possible future clinical applications. My friend and fellow student Arngrímur Einarsson I thank for reviewing a draft of this thesis. Last but not least I would like to thank my family, Jóhanna Magnúsdóttir and newborn Sindri, for support and taking on this journey with me of studying and living here in Copenhagen.

Contents

Chapter 1

Introduction

In resent years tremendous advances have been achieved in the ability to produce images of human brain function. Functional brain imaging is a multidisciplinary research field that encompasses techniques aimed to better understand the human brain through non-invasive imaging of the electrophysiological, hemodynamic, metabolic and neurochemical processes that underlie normal and pathological brain function. These techniques provide tools for non-invasive exploration of the brain both of interest for neuroscience research and clinical diagnosis of neurological and neuropsychological disorders, such as epilepsy, schizophrenia, depression, Parkinson's and Alzheimer's diseases.

Using radioactively labeled organic molecules, that are involved in processes of interest, brain metabolism and neurochemistry can be studied. Images of dynamic changes in the spatial distribution of these molecules can be formed using *positron emission tomography (PET)*. The spatial resolution is as high as 2mm while temporal resolution is limited to several minutes by the dynamics of the process being studied. *Functional magnetic resonance imaging (fMRI)* can be used to detect hemodynamic changes which can give more direct studies of neural activity. As neurons become active they induce very localized changes in blood flow and oxygenation. fMRI studies are capable of producing spatial resolution of 1mm and temporal resolution around 1s, which is better than PET but still poor compared to temporal dynamics of electrical neural activity.

Electroencephalography (EEG) measures electrical potentials on the scalp produced by electrical activity in neural cell assemblies. It directly measures electrical brain activity and provides superior temporal resolution to PET and fMRI, or in the range of milliseconds. However, the spatial resolution of EEG does not match that of PET and fMRI since it is limited by the small number of spatial scalp measurements and inherent ambiguity of the underlying electro-magnetic inverse problem. It is also important to note that even if the EEG measurements were continuous over the whole scalp the number of possible underlying source distributions is infinite. The inverse EEG problem is therefore ill-posed in nature and requires additional constraints and/or assumptions (e.g. anatomical) to be solved. The two general approaches to EEG source estimation are *equivalent dipole localization* or *parametric* and *distributed source imaging* or simply *imaging*. Equivalent dipole localization typically assumes that the sources can be represented by a few equivalent current dipoles of unknown location and moment to be estimated with a non-linear numerical method. Although this method gives good estimate when the number of active areas is small, it is difficult to determine the appropriate number of dipole sources for complicated spatio-temporal activity. The distributed source imaging assumes distributed currents in the brain volume. Often the sources are plausibly constrained to the cortical surface, thus a current dipole is assigned to each of many thousands of tessellation elements on the cortical surface. Since the locations are fixed only the orientation and amplitudes have to be determined, reducing the inverse problem to a linear one with strong similarities to those encountered in image reconstruction. Furthermore the dipole orientations are often constrained to equal the local surface normal, reducing the problem to only finding the am-

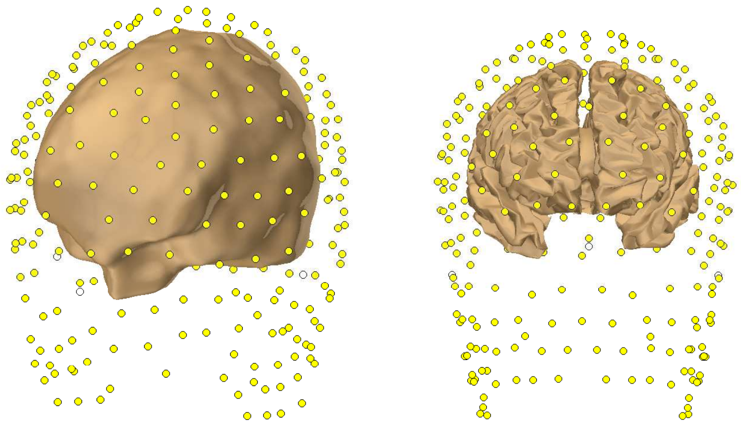

Figure 1.1: *256 EEG electrodes grid with respect to inner skull and brain.*

plitudes. In this thesis the focus is on the distributed source imaging method which is fundamentally ill-posed.

The ill-posedness of the inverse problem is most often tackled using regularization techniques, known from image restoration and reconstruction (e.g. see [1, 2] for extensive review). So called *Tikhonov regularizer* has two terms, one that measures the fit to the data and other that measures the smoothness of the image. Furthermore it has a regularization parameter that typically is chosen using cross-validation methods or the L-curve [1]. Other methods expand the classical Tikhonov regularizer by means of using different norms, e.g. 1-norm which presents sparsity to the solution, and constraints in the form of spatial smoothing or a more localized solution. Probably the most well known method is *LORETA (low resolution electromagnetic tomography)* [3, 4] which has the same form as the Tikhonov regularizer with an added smoothness constraint, in the form of a discrete Laplacian operator, and a normalization factor.

More recently a Bayesian approach to the inverse problem has been attempted [5, 6]. In its simplest form it is equivalent to the Tikhonov regularizer except that regularization parameters are optimized directly by means of iterative update formulas. Furthermore sparsity, spatial smoothing and possibly other constraints (anatomical, localized, etc.) can be incorporated into the prior distribution in a hierarchical fashion. In this thesis the inverse problem will be solved in a Bayesian fashion [7, 8, 9, 10, 11, 12] and different algorithms derived from the Bayesian framework. Examples will be presented both on artificial data, with different model complexities, and on real data. These examples should reveal pros and cons of different methods and lead to some general discussion and conclusions.

The concepts of *condition number* and *posedness* are important when describing and analyzing the inverse EEG problem and *signal-to-noise ratio* is crucial in all practical signal processing applications. Brief discussion about these concepts is therefore included in the next two sections.

1.1 Ill-conditioned vs. Ill-posed

Frequently in the text the terms *ill-conditioned* and *ill-posed* will be used when describing the inverse EEG problem. Although they often both apply an ill-

[1]The main idea of the L-curve method is to plot the smoothing norm as a function of the residual norm for all values of the regularization parameter. This curve has an L-shaped dependence and the optimal value of regularization parameter can be found at the corner of the L-curve.

conditioned problem is not necessarily ill-posed and vice versa so lets define these concepts to avoid confusion in the following text.

The term *well-posed* goes back to the French mathematician *J. S. Hadamard* (1865-1963) who believed that mathematical models of physical phenomena should have three properties, namely that a solution exists, the solution is unique and the solution depends continuously on the data. Examples of well-posed problems include the Dirichlet problem for Laplace's equation and the heat equation with specified initial conditions. By contrast the backwards heat equation, deducing a previous distribution of temperature from final data is not well-posed in that the solution is not unique and highly sensitive to changes in the final data. Problems that are not well-posed in the sense of Hadamard are termed *ill-posed*. The solution to the inverse EEG problem is never unique, thus the problem is ill-posed and requires some additional assumptions or constraints to be solved.

A well-posed problem may suffer from numerical instability when solved with finite precision on a computer or with errors on the data. So even if the problem is well-posed small changes in data can result in much larger errors in the answers, thus the problem being *ill-conditioned*. An ill-conditioned problem is indicated by a big condition number. In the case of a linear problem defined by a matrix $\mathbf{A}$ the condition number can be defined

$$\kappa = \frac{\max(\mathbf{d})}{\min(\mathbf{d})}, \tag{1.1}$$

where $\mathbf{d}$ is a vector containing the singular values of $\mathbf{A}$.

1.2 Signal-to-noise ratio (SNR)

Signal-to-noise ratio (SNR) is defined by the ratio of signal power to the noise power, ideally the noise should be zero giving infinite signal-to-noise ratio. But whenever one is working with measured signals there is some corruption of the signal. In electrical circuits there can be many sources for noise, e.g. *thermal noise* (also known as Johnson noise, Nyquist noise or white noise) is generated by the thermal agitation of the electrons inside an electrical conductor regardless of the applied voltage, 50Hz noise (60Hz in USA) is often detected in electrical devices because of the power grid supply voltage and in semiconductors one often detects *shot noise* due to energy barriers in pn-junctions, *generation-recombination noise* due to defects in the band gap and *1/f noise* whose cause is still under debate. Noise can also be generated by malfunction or bad handling of the equipment being used, e.g. in EEG measurements faulty or ill-placed electrodes can have added noise.

If we let P_s denote the signal power and P_n the noise power then the SNR is defined

$$\text{SNR} = \frac{P_s}{P_n}. \tag{1.2}$$

In electrical engineering SNR is often given in the units of decibels (dB), defined

$$\text{SNR}_{dB} = 10 \log \left(\frac{P_s}{P_n} \right).$$

where log is the base 10 logarithm. The SNR ratio can also be found from the signal amplitude, e.g. a resistor with resistance R dissipates the power V_s^2/R where V_s is the signal root mean square (rms) voltage. If there is noise present with rms voltage of V_n the signal to noise ratio can be written,

$$\text{SNR} = \frac{V_s^2/R}{V_n^2/R} = \frac{V_s^2}{V_n^2}$$

and equivalently in the units of dB

$$\text{SNR}_{dB} = 10 \log \left(\frac{V_s^2}{V_n^2} \right) = 20 \log \left(\frac{V_s}{V_n} \right).$$

For random signals the corresponding quantity to the power is the variance σ^2. Since we generally deal with stochastic variables here and the noise is assumed Gaussian distributed the signal-to-noise ratio is defined

$$\text{SNR} = \frac{\sigma_{m0}^2}{\sigma_\epsilon^2} \tag{1.3}$$

where σ_{m0}^2 and σ_ϵ^2 are the signal and noise variances respectively.

1.3 Thesis Overview

- Chapter 1 is an **Introduction** to the ill-posed EEG inverse problem. It includes a brief overview of previous work and methods used to model and solve the inverse problem.

- Chapter 2 is an introduction to **Electroencephalography (EEG)**. EEG basics and some brain anatomy are presented. For an experienced EEG veteran this chapter is not a necessary reading.

- Chapter 3 presents an important overview of the EEG forward problem. The EEG **Forward Model** is derived which is a very important part of solving the inverse problem. Here the physics of EEG are derived mathematically. Solutions for different head shape approximations are presented and an algebraic formulation of the forward problem shown, introducing the so-called lead field matrix.

- Chapter 4 is the main theory. **The Linear Inverse Problem** is presented and tackled using Bayes' theorem. First a simple Gaussian approach involving only two hyperparameters is derived. Then more complex prior distribution is used where the number of hyperparameters are of the same order as the number of sources. In this framework the author presents an original update algorithm which he has not found in the literature. This is one of the main parts of the thesis.

- Chapter 5 consists of **simulations** using artificial data for validation purposes. These simulations show how the theoretical work in chapter 4 progressed and what were the causes of trying different modifications and expansions of the algorithms.

- Chapter 6 shows the results of some **real data testing** using the algorithms.

- Chapter 7 is the **conclusion** of the work. This chapter summarizes the main results and gives a more clear overview of the simulations from chapter 5 in context with the theoretical work in chapter 4. Finally there are some notes regarding further work.

- The **Appendix** contains some additional information. In the mathematical appendix more details are provided on some of the derivations from the text. Proposed Matlab implementations of the algorithms are listed in detail. For reproducibility purposes detailed information is provided on the BEM head models used in the simulations and real data testing. Final part of the appendix presents some inverse solutions of sample data, this was not included in the main text due to lack of information on the data.

Chapter 2

Electroencephalography (EEG)

Neural activity in cell assemblies generates currents in the brain tissue causing a potential distribution throughout a subjects head resulting in easily measurable potential over the scalp. In *electroencephalography (EEG)* electrodes are placed on a subjects head to measure these potentials. EEG is practiced by all who are interested in the underlying neurophysiology, from medical professionals to scientists. The electrical potentials exhibit spatial and temporal patterns that depend on the nature and location of the sources. Since these dynamic patterns are correlated with behavior and cognition it has often been said that EEG provides a ”window on the mind”. German psychiatrist *Hans Berger* (1873-1941) is usually credited with the first scalp recordings of the human EEG in the mid-1920s. Others had performed similar experiments but he was the one who gave the device its name. This chapter will give some basic introductory information on EEG, starting with some human brain anatomy. The text is primarily based on a classic and widely acclaimed book by *Nunez and Srinivasan* [13], originally published in 1981 by the senior author but brought up to date and republished in 2006. Review article on biomagnetism by *Williamson and Kaufman* [14] provides a good overview and also the text by *Hämäläinen et al.* [15].

2.1 The Human Brain

The most complex structure known to exist is the human brain. In the outermost layer of the brain, the *cerebral cortex*, there are at least 10^{10} neurons which form a vast signal handling network of around 10^{14} interconnections or *synapses.* During information processing small currents flow in the neural system resulting in the measurable scalp potential. Figure 2.1 shows the human brain and some important anatomical features are identified. The three primary divisions of the

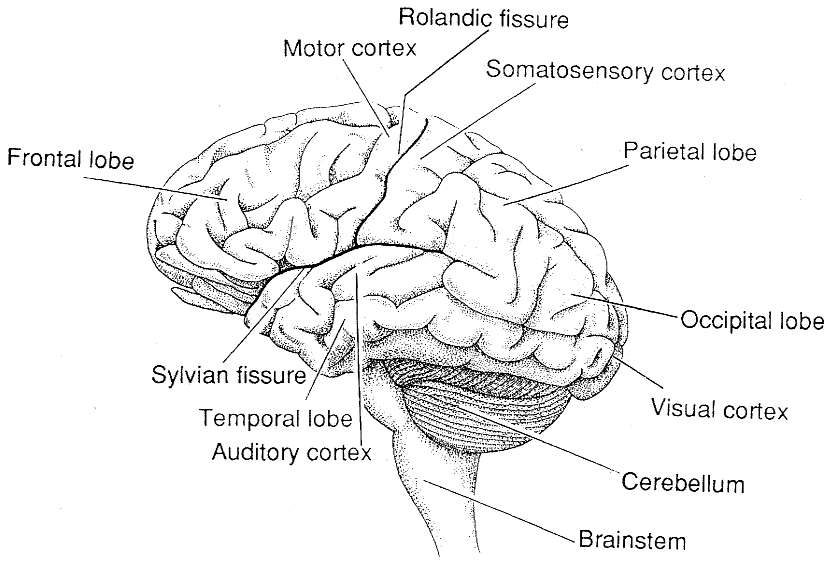

Figure 2.1: *Human brain with some important areas indicated. Figure adopted from [15]. Notice the motor cortex which is next to the Rolandic fissure stretching over both hemispheres. The motor cortex will be mentioned in the simulations and testing in chapters 5 and 6.*

human brain are the *brainstem* and *cerebellum*, which are marked on the figure, and *cerebrum* which is the large folded structure on the figure with many different areas indicated. The cerebrum consists of two hemispheres, left and right halves, separated by the longitudinal fissure. Each half is divided into lobes by two deep grooves, the Rolandic fissure runs down the side of both hemispheres while the Sylvian fissure is almost horizontal. There are four lobes in both halves of the cortex, namely the frontal lobe, parietal lobe, occipital lobe and the temporal lobe, all indicated on the figure. Most regions of the cortex have been mapped and few of them are marked on the figure, e.g. visual, auditory and motor cortex. EEG is usually mostly concerned with the cerebral cortex.

It is a 2 to 5 mm thick layer having a total surface are around 1600 to 4000 cm^2. Interconnections between neurons are very strong in this area, e.g. the surface of a large cortical neuron may be covered with as many as 10^5 synapses that transmit inputs, known as *action potentials*, from other neurons. The inputs to a neuron are of two types, *excitatory postsynaptic potentials (EPSPs)* and *inhibitory postsynaptic potentials (IPSPs)* which make it easier and harder respectively for the neuron to fire an action potential. EPSPs produce local membrane current sinks with corresponding distributed passive sources. IPSPs produce local current sources with more distant distributed passive sinks. Our consciousness must involve, in some mostly unknown manner, the interaction of cortical neurons.

The cortex tissue is called *gray matter*. When alive it is actually pink but when stained by anatomists post mortem the cell bodies turn gray. *White matter* is just below the gray matter and consists of numerous interconnections between cortical regions. Hundreds of substructures within the brain have been labeled by anatomists but here we are interested in larger structures near the surface that are capable of generating potentials sufficiently coherent to be observed on the scalp.

Neural Basics

Neurons and *glial cells* are the principal building blocks of the brain. The glial cells are important for structural support, ionic concentration maintenance and for transportation of nutrients and other substances between blood vessels and brain tissue. Neurons are the information processing units with their cell bodies concentrated in the gray matter. A neuron consists of the *soma* (cell body), the *dendrites* and the *axon* as shown on figure 2.2. The soma contains the nucleus and much of the metabolic machinery. The dendrites are threadlike extensions that receive stimuli from other cells and the axon is a single long fiber that carries the nerve impulse away from the soma to other cells. *Pyramidal* and *stellate* cells are the two principal groups of cortical neurons where the pyramidal ones are relatively larger. Their dendrites tend to be perpendicular to the cortical surface, resulting in neural currents being perpendicular to the cortical sheet of gray matter. This is an important property which will be used in chapter 4 to simplify the forward model. Dendrites typically have thousands of connections (synapses) from other neurons. Excitatory synapses are most often on the dendrites and inhibitory synapses often attach to the soma.

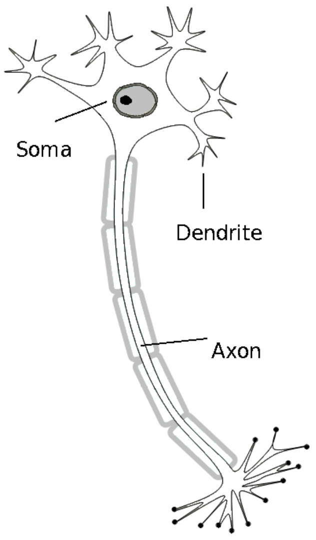

Figure 2.2: *Schematic of a neuron with its three main parts indicated.*

2.2 EEG Basics

Dynamic brain behaviour is believed to result from the interaction of neurons and assemblies of neurons at multiple spatial scales. EEG electrodes can measure part of the dynamic behavior at macroscopic scales. This electrical activity is divided into two major categories, namely *spontaneous potentials* such as sleep rhythms and *evoked potentials (EPs)* or *event related potentials (ERPs)* which are direct response to external stimulus such as an auditory tone or a light flash. Using repeated stimulus, such as light flashes, a time averaged EP can be calculated to remove the spontaneous EEG. The theoretical work in later chapters deals with EEG in general, i.e. independently of which type of activity is considered. But in the artificial simulations, chapter 5, and the real data testing, chapter 6, the examples apply to EPs and ERPs. In most of the artificial simulations we deal with single pulses on a cortical surface, these simulations correspond to ideal averaged EPs where all background activity has been averaged out.

Table 2.1 lists classification of different frequency bands for EEG measurements. This classification is based on early experiments and findings. When pioneers were interpreting their early results spectral analysis (Fourier transform) was not in use. EEG waveforms were characterized by visual inspection and zero crossings. This procedure tends to emphasize faster frequencies and it is not

Range	**Rhythm**
1 - 4 Hz	delta
4 - 8 Hz	theta
8 - 13 Hz	alpha
13 - 20 Hz	beta
$>$ 20 Hz	gamma

Table 2.1: *EEG frequency domains classification. These values are from [13], there is however some inconsistency in the literature specially regarding the beta and gamma frequency ranges, e.g. the author has seen the beta range classified from 13 to 30 Hz, 13 to 22 Hz and from 12 to 26 Hz.*

clear to what extend overlap between regions occurred in early recordings. Frequency ranges in the table are only roughly divided and inconsistency in the literature is common. Underlying physiological processes have been linked to different rhythms. Delta rhythms appear during sleep and in babies in the first few months of age, its amplitude increases with eye closure and is believed to be a precursor of mature alpha rhythms. The alpha rhythm is considered the main EEG rhythm and the amplitude of scalp alpha oscillations is typically 20 to 50 μV. Normal resting alpha rhythms may be reduced in amplitude by eye opening, drowsiness and challenging mental tasks. Like most EEG phenomena the alpha rhythms exhibit inverse relationship between amplitude and frequency, the physiological base for this inverse relation is largely unknown. For clinical EEG examination alpha rhythms provide an appropriate starting point. During periods of mental activity beta rhythms appear.

Electrode Placement

The *international 10-20 system* is an EEG standard used for placing electrodes on a subjects head, originally proposed in 1958. Since then it has become the standard for clinical as well as non-clinical EEG. On figure 2.3 the 10-20 system is shown on a simplified 2D head viewed from above with the nose indicated at the top. However advancement in EEG studies has lead to multi-channel EEG hardware systems with much larger number of electrodes. Extensions of the 10-20 system have therefore been proposed with up to 345 electrode positions [17]. The placement of the electrodes is based on landmarks on the skull, namely the nasion (Nz), the inion (Iz) and the left and right preauricular points (LPA and RPA, placed near the ear canals). With respect to the 2D map on figure 2.3 these landmarks would be placed at the top, the bottom and left and right respectively. Contours and lines are then formed between these landmarks and the numbers 10 and 20 indicate percentages of the total distances

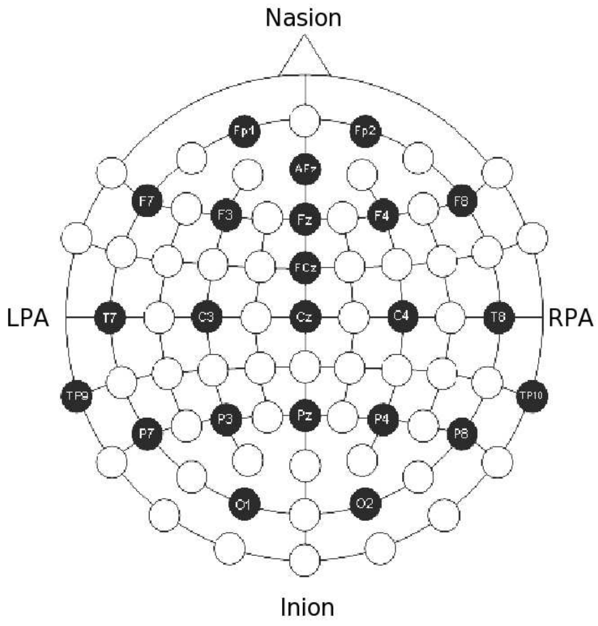

Figure 2.3: *International 10-20 System recording cap available from EasyCap [19]. 19 or 21 channels of the black labeled dots are used for recording and FCz is recommended for reference and AFz for ground. The additional unmarked circles are positions used for a higher density electrode grids.*

of these landmark lines. This is best explained by an example. Lets look at the line from LPA to RPA which runs through T7, C3, Cz, C4 and T8. We shall denote the total length of the line by L, then T7 is placed 10% of L from the LPA landmark, C3 is then placed 20% of L from T7, Cz is placed 20% of L from C3 and so forth until T8 is reached which is placed 10% of L from RPA and 20% from C4. Extended systems place the electrodes more densely on the scalp, thus the distances between electrodes are shorter. Instead of 10 and 20% lower percentages of the total distance are used with values down to 5 and 10%. These systems are therfore called 10-10 and 5-10, where 5-10 is the most densely packed with up to 345 defined locations [17].

Chapter 3

Forward Model

Here the physics of EEG will be modeled. Surface integral equations will be derived using a well known approximation of Maxwell's equations. This enables forward calculations of EEG potentials given a specific set of neural current sources. The head shape plays a role when solving the surface integrals. Different head shape approximations will be introduced and algebraic formulation of the forward model presented. The forward model is necessary to be able to tackle the inverse probelm. One can therefore say that this chapter is the first step in solving the inverse EEG problem. The surface integral derivation in this chapter is mostly based on papers by *Geselowitz* [20, 21]. Overview texts with some nice explanations can also be found in [1, 14, 15, 22]. Good discussion on different head models can be found in [23, 24]. The final part of the chapter is an introduction to the forward model software package used [25].

3.1 Current Distribution and Dipoles

In the ideal case a sensory stimulus activates a small portion of the cortex which causes measurable electric potentials on the scalp. This activation in the cortex can be linked to membrane-spanning ionic flow in the neurons where chemical energy is converted to electrical potential over the cell membrane. This membrane ionic flow shall be called *impressed current* or *primary current* $\mathbf{J}^p(\mathbf{r})$

and has the unit of ampere per square meter. As a passive response to gradients of the electrical potential set up by impressed currents another current appears in the surrounding tissue. This current flow shall be called the *volume current* $\mathbf{J}^v(\mathbf{r})$. Volume currents therefore represent the movement of charge in surrounding tissue dictated by conductivity $\sigma(\mathbf{r})$ and electric field $\mathbf{E}(\mathbf{r})$. Ohm's law gives $\mathbf{J}^v(\mathbf{r}) = \sigma(\mathbf{r})\mathbf{E}(\mathbf{r})$ and the total current at each point $\mathbf{r}$ in space is $\mathbf{J}(\mathbf{r}) = \mathbf{J}^p(\mathbf{r}) + \mathbf{J}^v(\mathbf{r})$. Our interest is the impressed current which represents the brain activity and below we will derive surface integral equations to calculate scalp potentials from the the impressed currents.

Just as the magnetic dipole serves as the elemental generating source in magnetism then in electrophysiology the concept of a discrete current dipole $\mathbf{q}$ has proven useful as the elemental generating source when modeling impressed currents. This can be thought of as the movement of a localized charge over a very short distance, where the product of current and distance gives the moment $\mathbf{q}$ of the current dipole and the direction coincides with that of the current. The unit of $\mathbf{q}$ is ampere-meter. Thinking in terms of discretized points in space, $\mathbf{q}$ at $\mathbf{r}_q$ can also be thought of as the concentration of the impressed current to a single point, i.e.

$$\mathbf{J}^p(\mathbf{r}) = \mathbf{q}\delta(\mathbf{r} - \mathbf{r}_q) \tag{3.1}$$

where $\delta(\mathbf{r})$ is the Dirac delta function. One can say that the current dipole is a straightforward extension of the more well known model of paired-charges dipole in electrostatics. It is important to emphasise that the brain activity does not actually consist of discrete sets of physical current dipoles, but rather that the dipole is a convenient representation for coherent activation of a large number of pyramidal cells.

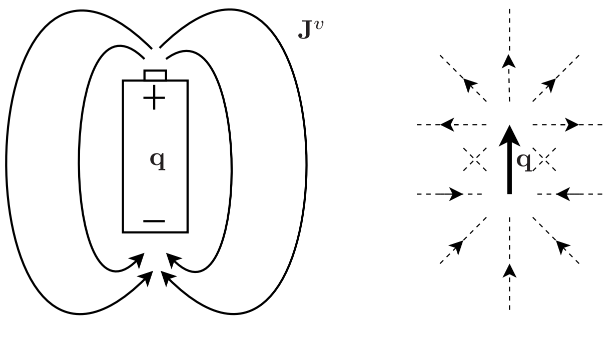

Figure 3.1: *On the left a current dipole is viewed as a battery immersed in a conducting medium. The battery represents the dipole moment and the surrounding backflow current is the volume current density. On the right the current dipole is viewed as a point source and a nearby point sink.*

Williamson and Kaufman [14] present a good intuitive example of how one can think of the current dipole approximation. A current dipole can be viewed as a small battery. Inside it biochemical processes impress a specified current directly from the negative to the positive terminal. Immersing this current dipole in a conducting medium a backflow is generated outside the battery described by the volume current density $\mathbf{J}^v(\mathbf{r})$. This example is illustrated on figure 3.1. Equivalently, the volume current can be viewed as the superposition of a radially symmetric outflow from a point source and an equal radially symmetric inflow toward a nearby point sink. This is also illustrated on figure 3.1.

3.2 Forward Model Integral Equations

From previous section we know that the total current density at each point in the head volume can be divided into two components, i.e. the primary current flow $\mathbf{J}^p(\mathbf{r})$ and the volume current $\mathbf{J}^v(\mathbf{r})$. Since EEG studies generally deal with frequencies on the range from 0.1Hz to 100Hz the physics can be described by using the quasi-static approximation of Maxwell's equations [21]. That means that the electric field can be expressed with a scalar potential $V(\mathbf{r})$

$$\mathbf{E}(\mathbf{r}) = -\nabla V(\mathbf{r}) \tag{3.2}$$

and then the current can be written as

$$\mathbf{J}(\mathbf{r}) = \mathbf{J}^p(\mathbf{r}) - \sigma(\mathbf{r})\nabla V(\mathbf{r}). \tag{3.3}$$

Neglecting tissue capacitance is reasonable for the frequency range mentioned above, that gives

$$\nabla \cdot \mathbf{J} = 0. \tag{3.4}$$

Combining this with equation 3.3 gives

$$\nabla \cdot \mathbf{J}^p(\mathbf{r}) = \nabla \cdot \sigma(\mathbf{r})\nabla V(\mathbf{r}). \tag{3.5}$$

This differential equations fully describes the relation beween the primary currents and voltages throughout the head, i.e. it formulates the forward problem. However it is not feasible to solve and we are only interested in the scalp surface potentials. The rest of this section therefore derives a surface integral equation relating the surface potentials with the primary currents.

We continue by looking at Green's second identity, as proposed by *Geselowitz* [20]. It states for scalar fields ϕ and ψ and surface S bounding volume G that

$$\int_G (\phi\nabla^2\psi - \psi\nabla^2\phi)dv = \int_S (\phi\nabla\psi - \psi\nabla\phi) \cdot d\mathbf{s}$$

By appropriately identifying the scalar fields Green's identity can give the desired integral equations. We assume the head consists of different regions (brain, skull, scalp, etc.) of uniform and isotropic conductances σ_i'. Then using the well behaved functions $\phi = \frac{1}{|\mathbf{r}-\mathbf{r}_q|}$ and $\psi = V$, as Geselowitz suggests, along with Green's identity gives

$$\sum_i \sigma_i \int_{G_i} \left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|} \nabla^2 V - V \nabla^2 \left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|} \right) \right) dv =$$

$$\sum_i \int_{S_i} \left(\sigma_i' \left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|} \nabla V' - V' \nabla \frac{1}{|\mathbf{r}-\mathbf{r}_q|} \right) - \right.$$

$$\left. \sigma_i'' \left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|} \nabla V'' - V'' \nabla \frac{1}{|\mathbf{r}-\mathbf{r}_q|} \right) \right) \cdot d\mathbf{s}_i,$$

where σ_i' and σ_i'' are the conductivities of the inner and outer sides respectively of surface S_i bounding volume G_i, as illustrated on figure 3.2. Remember that $\mathbf{r}$ refers to the voltage location and $\mathbf{r}_q$ to the dipole location. From here on we assume that $d\mathbf{s}_i$ is directed from the primed region to the double primed. And at the external boundary $\sigma'' = 0$ is assumed, i.e. zero air conductivity. Now

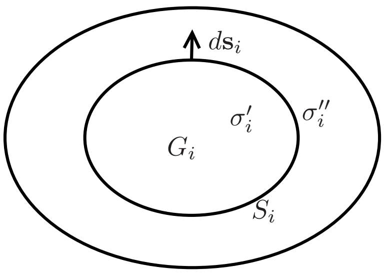

Figure 3.2: *Surface S_i, bounding volume G_i, with inner conductivity σ' and outer conductivity σ''.*

using boundary conditions of continuous voltages and currents on an interface between regions of conductivities σ' and σ'', i.e. [1]

$$V' = V'' \quad \text{and} \quad \sigma' \delta V'/\delta n = \sigma'' \delta V''/\delta n \quad \text{on } S_i,$$

the right hand side of the equation reduces to

$$-\sum_i \int_{S_i} \left((\sigma_i' - \sigma_i'') V \nabla \left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|} \right) \right) \cdot d\mathbf{s}$$

On the left hand side the following relationship can be used

$$\nabla^2 \left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|} \right) = -4\pi \delta(\mathbf{r}-\mathbf{r}_q)$$

[1]regarding the notation then $\delta V/\delta n = \nabla V \cdot d\mathbf{s}$

where $\delta(\mathbf{r})$ is the Dirac delta function[2]. This gives

$$\sum_i \sigma_i \int_{G_i} \left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|}\right) \nabla^2 V dv + V(\mathbf{r}) 4\pi\sigma_i(\mathbf{r}) = -\sum_i \int_{S_i} (\sigma_i' - \sigma_i'') V \nabla \left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|}\right) \cdot d\mathbf{s}$$

Putting the identity from equation 3.5 into the equation above and assuming all primary currents are confined within a single homogeneous volume, namely the brain, we get

$$\int_G \frac{\nabla \cdot \mathbf{J}^p}{|\mathbf{r}-\mathbf{r}_q|} dv + V(\mathbf{r}) 4\pi\sigma = -\sum_i \int_{S_i} (\sigma_i' - \sigma_i'') V \nabla \left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|}\right) \cdot d\mathbf{s}$$

or

$$\sigma V(\mathbf{r}) = \sigma_0 V_0(\mathbf{r}) - \frac{1}{4\pi} \sum_i (\sigma_i' - \sigma_i'') \int_{S_i} V(\mathbf{r}_q) \nabla \left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|}\right) \cdot d\mathbf{s} \tag{3.6}$$

where the primary potential is

$$V_0(\mathbf{r}) = -\frac{1}{4\pi\sigma_0} \int_G \frac{\nabla \cdot \mathbf{J}^p}{|\mathbf{r}-\mathbf{r}_q|} dv$$

Here σ_0 is needed to get the dimensions right and V_0 is the potential due to $\mathbf{J}^p$ in an infinite homogeneous medium with unit conductivity. The integral in the equation for V_0 can be transformed using the following

$$\begin{aligned} \int_G \nabla \cdot \left(\frac{\mathbf{J}^p}{|\mathbf{r}-\mathbf{r}_q|}\right) dv &= \int_S \frac{\mathbf{J}^p}{|\mathbf{r}-\mathbf{r}_q|} \cdot d\mathbf{s} \\ &= \int_G \left(\mathbf{J}^p \cdot \nabla \left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|}\right) + \left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|}\right) \nabla \cdot \mathbf{J}^p\right) dv \end{aligned}$$

and assuming $\mathbf{J}^p$ vanishes on the boundary of the region containing the sources then

$$\int_G \left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|}\right) \nabla \cdot \mathbf{J}^p dv = -\int_G \mathbf{J}^p \cdot \nabla \left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|}\right) dv.$$

Putting this into the equation for $V_0(\mathbf{r})$ gives

$$V_0(\mathbf{r}) = \frac{1}{4\pi\sigma_0} \int_G \mathbf{J}^p \cdot \nabla \left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|}\right) dv. \tag{3.7}$$

In equation 3.6 above the placement of $\mathbf{r}$ has not been specified, i.e. the voltage can be at an arbitrary place. To obtain an integral equation for $V(\mathbf{r})$ on the surfaces S_i we let $\mathbf{r}$ approach a point on S_i from the inside (where the conductivity is σ'). *Vladimirov* [29] derives a limit rule for this case which can be used

[2]e.g. see Vladimirov [29] and/or Strauss [30] for more details on this identity

here [3]. Let $F_i(\mathbf{r})$ represent the integral on the right hand side of equation 3.6, i.e.

$$F_i(\mathbf{r}) = \int_{S_i} V(\mathbf{r}_q)\nabla\left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|}\right)\cdot d\mathbf{s}.$$

If we let $\mathbf{r}_0$ be a point in the region inside of surface S_i and let $\mathbf{r}_0$ approach a point, $\mathbf{r}$, on the surface S_i ($\mathbf{r} \in S_i$) then the limit rule from Vladimirov states

$$\lim_{\mathbf{r}_0\to\mathbf{r}\in S_i} F_i(\mathbf{r}_0) = -2\pi V(\mathbf{r}) + F_i(\mathbf{r}).$$

Using this on the integral in equation 3.6 gives for $\mathbf{r}$, on some boundary S_k,

$$\sigma' V(\mathbf{r}) - \frac{1}{2}(\sigma' - \sigma'') = \sigma_0 V_0(\mathbf{r}) + \frac{1}{4\pi}\sum_i(\sigma_i' - \sigma_i'')\int_{S_i} V(\mathbf{r}_q)\frac{\mathbf{r}-\mathbf{r}_q}{|\mathbf{r}-\mathbf{r}_q|^3}\cdot d\mathbf{s}$$

or equivalently an integral equation for $\mathbf{r} \in S_k$

$$(\sigma' + \sigma'')V(\mathbf{r}) = 2\sigma_0 V_0(\mathbf{r}) + \frac{1}{2\pi}\sum_i(\sigma_i' - \sigma_i'')\int_{S_i} V(\mathbf{r}_q)\frac{\mathbf{r}-\mathbf{r}_q}{|\mathbf{r}-\mathbf{r}_q|^3}\cdot d\mathbf{s} \qquad (3.8)$$

where S_k is the scalp in the case of EEG measurements (note that $\frac{\mathbf{r}-\mathbf{r}_q}{|\mathbf{r}-\mathbf{r}_q|^3} = -\nabla\left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|}\right)$).

Now to briefly summarize the results for the forward calculations. If one knows the current density, $\mathbf{J}^p$, equation 3.7 can be used to calculate the primary potential V_0. Which can then be used to solve 3.8 for the potential $V(\mathbf{r})$ on surface S_k, which would be the scalp in the case of EEG measurements.

A related measurement technique called *magnetoencephalography (MEG)* measures the magnetic field, $\mathbf{B}(\mathbf{r})$, outside the head volume using multichannel SQUID (superconducting quantum interference device) gradiometers. Same methods as used for EEG are used when solving the MEG inverse problem. These two techniques are complementary and often used together but in this thesis the focus is only on EEG. However in appendix A.2 the forward model MEG integral equations are derived and interestingly when solving the MEG forward problem one must also solve the EEG forward problem.

[3] see pages 291-303 in Vladimirov [29] for more details on the derivation of this limit rule

3.3 Algebraic Formulation

Having derived the integral equations for solving the forward problem algebraic equations can be introduced that describe the scalp potential $V(\mathbf{r})$. In the following text the measured scalp EEG potential shall be noted by $m(\mathbf{r})$, i.e. when $\mathbf{r}$ is a point on the scalp then $V(\mathbf{r}) = m(\mathbf{r})$. It can be shown that the scalp potential measurements are linear with respect to the dipole moment $\mathbf{q}$ and non-linear with respect to the location $\mathbf{r}_q$ of the dipole [15, 23]. For reasons that will be clear soon it is convenient to separate the dipole magnitude $q = |\mathbf{q}|$ and orientation $\theta = \mathbf{q}/|\mathbf{q}|$. The scalp electric potential at location $\mathbf{r}$ generated by a single dipole can then be written

$$m(\mathbf{r}) = a(\mathbf{r}, \mathbf{r}_q, \theta)q \tag{3.9}$$

where $a(\mathbf{r}, \mathbf{r}_q, \theta)$ is the solution to the electric forward problem for a dipole with unit amplitude and orientation θ. Expanding this by linear superposition to simultaneous activation of P dipoles located at $\mathbf{r}_{qi}$ $(i = 1, ..., P)$ and N scalp measurements at $\mathbf{r}_j$ $(j = 1, ..., N)$ gives

$$\mathbf{m} = \begin{bmatrix} m(\mathbf{r}_1) \\ \vdots \\ m(\mathbf{r}_N) \end{bmatrix} = \begin{bmatrix} a(\mathbf{r}_1, \mathbf{r}_{q1}, \theta_1) & \dots & a(\mathbf{r}_1, \mathbf{r}_{qP}, \theta_P) \\ \vdots & \ddots & \vdots \\ a(\mathbf{r}_N, \mathbf{r}_{q1}, \theta_1) & \dots & a(\mathbf{r}_N, \mathbf{r}_{qP}, \theta_P) \end{bmatrix} \begin{bmatrix} q_1 \\ \vdots \\ q_P \end{bmatrix} \tag{3.10}$$

or simply

$$\mathbf{m} = \mathbf{A}\mathbf{s}, \tag{3.11}$$

where $\mathbf{A}$ has dimensions $N \times P$ and for EEG scalp recordings $N << P$, making the inverse solution very ill-posed. $\mathbf{A}$ is often called the *lead field matrix* or simply the *gain matrix* and $\mathbf{s} = [q_1, q_2, ..., q_P]^T$ is a vector of dipole magnitudes. In later chapers $\mathbf{s}$ will also be refered to as simply the source vector and denoted $\mathbf{s} = [s_1, s_2, ..., s_P]^T$. This model can be extended to include a time component t when time evolving activities of each dipole are considered. The orientation gain matrix $\mathbf{A}$ can therefore be calculated for each time index allowing the dipole to rotate as a function of time.

Alternatively $\mathbf{A}$ can be fixed allowing the addition of discrete time samples to the model. Then for P sources and N sensors at T discrete time samples the spatio-temporal model can be represented as

$$\mathbf{M} = \begin{bmatrix} m(\mathbf{r}_1, 1) & \dots & m(\mathbf{r}_1, T) \\ \vdots & \ddots & \vdots \\ m(\mathbf{r}_N, 1) & \dots & m(\mathbf{r}_N, T) \end{bmatrix} = \mathbf{A} \begin{bmatrix} q_{11} & \dots & q_{1T} \\ \vdots & \ddots & \vdots \\ q_{P1} & \dots & q_{PT} \end{bmatrix} = \mathbf{A}\mathbf{S}^T \tag{3.12}$$

where $\mathbf{S}^T$ is a $P \times T$ spatio-temporal matrix of dipole amplitudes q_{ij} $(i = 1, ..., P$ and $j = 1, ..., T)$ and $\mathbf{M}$ is the spatio-temporal $N \times T$ matrix of EEG scalp potentials.

3.4 Head Models

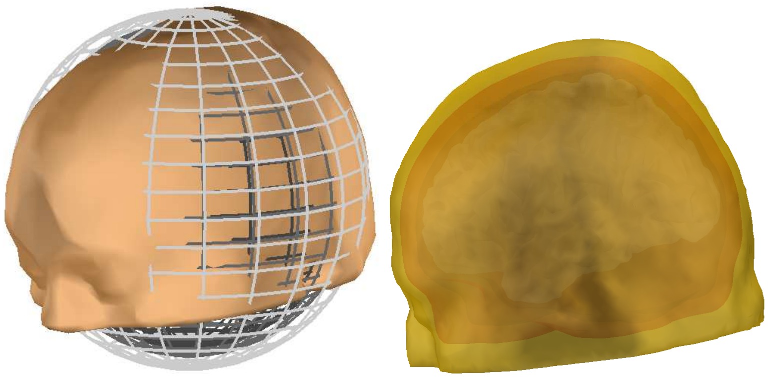

Figure 3.3: *Figure on the left shows three concentric spheres fitted to head tesselation surfaces. Parts of the frontal cortex are cut off. On the right tesselation surfaces extracted from anatomical data are shown.*

To solve the forward problem described by equation 3.8 one needs to know the structure of the surfaces for the different volumes of the head. Analytic solutions exist of equation 3.8 for simple head models in the form of spherical models. Because of their simplicity and ease of computation they have traditionally been used for approximating the human head. There are however some fundamental drawbacks which lie in its shape. Sensor positions need to be projected onto the fitted sphere, distorting the true sensor-dipole spatial geometry. In the case of single multilayer spheres there is incomplete coverage of brain areas, typically ignoring the frontal cortex, as figure 3.3 illustrates. Compensating for these kinds of errors is often resolved by fitting additional spheres to those regions not covered by the primary sphere, complicating the EEG forward model since neural sources may simultaneously be inside of some spheres while outside others.

The head shape is clearly not spherical and improvements can be made by using a more realistic head shape. High resolution spatial information is generally extracted from anatomical images (e.g. using MRI or CT scans) giving surface tesselations of the scalp, outer skull, inner skull and other regions. The brain tesselation surface can furthermore be used to confine sources on or within the brain surface. For a surface of arbitrary shape analytical solutions for the potentials over multilayer surfaces do not exist. Using numerical methods such as the *boundary element method (BEM)* or other related techniques the surface integral equations can be solved to find the surface potentials. The major drawback of BEM and related numerical methods is their computational cost, exceeding that of spherical models by orders of magnitude.

In this section different head models will be discussed [23, 24] and in a following section the *BrainStorm* software package [25], which was used for the forward modeling, will be introduced.

3.4.1 Spherical

Single-layer spherical shell with uniform conductivity is probably the simplest EEG head model. A closed form solution was introduced in 1973 but in practice this model proves far too simplistic for the human head which consists of multiple layers where the conductivity varies as much as two orders of magnitude between skull and brain (as shown in table table 3.1 on page 26). To account for this great variety in conductance analytic solutions have been derived for *three- and four-layer concentric spheres*, using anatomically extracted tesselation surfaces of brain, skull, scalp and sometimes *cerebrospinal fluid (CSF).* Multilayer spherical models are by far the most widely used because of their simplicity and relatively good accuracy. The closed form solution for the single sphere is compact and straight forward but the multishell case requires the evaluation of an infinite series. Methods to improve the computational efficiency of multilayer spherical models have focused primarily on approximating the infinite series. Here we will start by presenting a closed form solution for the single sphere following a discussion on multishell models and single sphere based approximations of the multishell model.

Single Sphere

Referring to the geometry of figure 3.4 where $\mathbf{r}$ and $\mathbf{r}_q$ are vectors pointing to an electrode position on the boundary and dipole within the volume respectively. The dipole is represented by $\mathbf{q}$ as before and $\mathbf{d} = \mathbf{r} - \mathbf{r}_q$ is introduced to make the equations below more compact. Three angles are additionally defined, ϕ_1 is the angle between the vector pointing to surface position $\mathbf{r}$ and dipole location $\mathbf{r}_q$, ϕ_2 is the angle that the dipole makes with the radial direction at $\mathbf{r}_q$ and ϕ_3 is the angle between the plane formed by $\mathbf{r}_q$ and $\mathbf{q}$ and the plane formed by $\mathbf{r}_q$ and $\mathbf{r}$. The signed dipole intensity can be represented by its radial and tangential components, i.e. $q_r = q \cos \phi_2$ and $q_t = q \sin \phi_2$ respectively The measured potential $V^1(\mathbf{r})$ on the surface at location $\mathbf{r}$ can then be expressed as a sum of two potentials

$$V^1(\mathbf{r}) = V_r(\mathbf{r}) + V_t(\mathbf{r}) \tag{3.13}$$

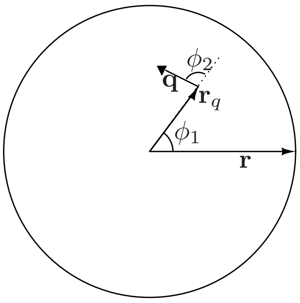

Figure 3.4: *Geometries for a single sphere model with uniform conductivity σ. The angle ϕ_3 is not shown on this 2D figure. It is the angle between the plane formed by $\mathbf{r}_q$ and $\mathbf{q}$ and the plane formed by $\mathbf{r}_q$ and $\mathbf{r}$.*

where

$$V_r(\mathbf{r}) = \left(\frac{q_r}{4\pi\sigma}\right)\left(\frac{2(r\cos\phi_1 - r_q)}{d^3} + \frac{1}{r_q d} - \frac{1}{r r_q}\right) \tag{3.14}$$

and

$$V_t(\mathbf{r}) = \left(\frac{q_t}{4\pi\sigma}\right)\cos\phi_3 \sin\phi_1 \left(\frac{2r}{d^3} + \frac{d+r}{rd(r - r_q\cos\phi_1 + d)}\right). \tag{3.15}$$

As was mentioned in the algebraic formulation section before, these equations show that the measured voltages are linear with respect to the dipole moment $\mathbf{q}$ but non-linear with respect to the dipole location $\mathbf{r_q}$.

Multishell Sphere

To account for the large conductivity differences of brain and skull multishell spherical models typically include three layers for the brain, skull and scalp. Sometimes additional CSF layer is included. The multishell case of M spherical shells requires the evaluation of an infinite series, namely

$$\begin{aligned} V^M(\mathbf{r}) &= \frac{q}{4\pi\sigma_M r^2}\sum_{n=1}^{\infty}\frac{2n+1}{n}\left(\frac{r_q}{r}\right)^{n-1} \ldots \\ & \quad f_n(n\cos\phi_2 P_n(\cos\phi_1) + \cos\phi_3\sin\phi_2 P_n^1(\cos\phi_1)) \end{aligned} \tag{3.16}$$

where P_n and P_n^1 are the Legendre and associated Legendre polynomials[4] respectively and

$$f_n = \frac{n}{nm_{22} + (1+n)m_{21}}. \tag{3.17}$$

[4]in problems with spherical symmetry Legendre's differential equation is often encountered, for more details see e.g. [30].

The m_{22} and m_{21} coefficients are found from

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} = \frac{1}{(2n+1)^{M-1}} \prod_{k=1}^{M-1} \cdots$$
$$\begin{bmatrix} n + \frac{(n+1)\sigma_k}{\sigma_{k+1}} & (n+1)\left(\frac{\sigma_k}{\sigma_{k+1}} - 1\right)\left(\frac{r_q}{r_k}\right)^{2n+1} \\ n\left(\frac{\sigma_k}{\sigma_{k+1}} - 1\right)\left(\frac{r_k}{r_q}\right)^{2n+1} & (n+1) + \frac{n\sigma_k}{\sigma_{k+1}} \end{bmatrix} \tag{3.18}$$

where the conductivities are arranged from the innermost sphere to the outermost, $\sigma_1, ..., \sigma_M$, corresponding to the radii of the spheres $r_1 < ... < r_M$. The matrices in equation 3.18 are non-commuting with the highest index matrix applied first. Although this is a closed form solution it includes and infinite series which has to be truncated or approximated. Under certain circumstances a single-sphere model can approximate a three-sphere model with good accuracy. Using the notation above, with $\mathbf{r}_q$ and $\mathbf{q}$ added to the dependent variables, $V^1(\mathbf{r}, \mathbf{r}_q, \mathbf{q})$ represents the single-sphere and $V^3(\mathbf{r}, \mathbf{r}_q, \mathbf{q})$ represents the three-sphere model. Then the approximation can be written as

$$V^3(\mathbf{r}, \mathbf{r}_q, \mathbf{q}) \simeq \lambda V^1(\mathbf{r}, \mu\mathbf{r}_q, \mathbf{q}), \tag{3.19}$$

where $V^3(\mathbf{r}, \mathbf{r}_q, \mathbf{q})$ is approximated by adjusting the location of the dipole along its radial direction by a scaling factor μ, then computing the much simpler single-sphere solution scaled by a factor λ. Exceptionally good approximations have been created in this way for three- and four-sphere head models and one commonly used is called the *Berg approximation*.

Overlapping Spheres

Overlapping spheres or *sensor fitted sphere* method fits a multishell sphere individually to each EEG sensor. This resolves the problem of ignoring brain areas as was illustrated on figure 3.3 when using multishell spheres. But there is some extensive added complexity which results in computational cost equivalent to that of using BEM models. We therefore focus on the BEM method in this text and introduce it in the next section. A detailed discussion of overlapping sphere head models can be found in [24].

3.4.2 BEM

To account for the non-spherical shape of the head the boundary element method can be used by using anatomically extracted shapes. The boundary element method is a numerical computational method of solving linear partial differential equations which have been formulated as integral equations, i.e. in boundary integral form as equation 3.8. Here a short introduction to the BEM approach using the *method of weighted residuals* will be presented. A detailed review of this method can be found in [23].

The forward problem from equation 3.8 can be rewritten as

$$\sigma_0 V_0(\mathbf{r}) = \frac{(\sigma' + \sigma'')}{2} V(\mathbf{r}) - \frac{1}{4\pi} \sum_i (\sigma_i' - \sigma_i'') \int_{S_i} V(\mathbf{r}') \frac{\mathbf{r} - \mathbf{r}'}{|\mathbf{r} - \mathbf{r}'|^3} \cdot d\mathbf{s} \tag{3.20}$$

and the right hand side of this equation can be expressed as a linear operator acting on the potential function $V(\mathbf{r})$, i.e.

$$\sigma_0 V_0(\mathbf{r}) = L(V(\mathbf{r})). \tag{3.21}$$

The source is known in the forward problem and hence the function $V_0(\mathbf{r})$ is known. The task is therefore to determine $V(\mathbf{r})$ so that the residual $L(V(\mathbf{r})) - \sigma_0 V_0(\mathbf{r})$ is as small as possible. Using the method of weighted residuals solves this problem using a weighting function $w(\mathbf{r})$, i.e. solving the related problem

$$\int_S (L(V(\mathbf{r}')) - \sigma_0 V_0(\mathbf{r}'))\, w(\mathbf{r}') d\mathbf{r}' = 0 \tag{3.22}$$

where the integration is over the domain of the unknown potential, namely the surface S. If (f, g) denotes the inner product of two functions f and g the equation can equivalently be written

$$(w(\mathbf{r}), V_0(\mathbf{r})) = (w(\mathbf{r}), L(V(\mathbf{r}))) \tag{3.23}$$

The selection of a particular weighting function determines the class of error method. In the BEM the weighting functions are restricted to a finite combination of N known linearly independent basis functions $\psi_n(\mathbf{r})$,

$$w(\mathbf{r}) = \sum_{n=1}^{N} \chi_n \psi_n(\mathbf{r}). \tag{3.24}$$

Coefficients χ_n are arbitrary, such that w spans this N dimensional space. Equation 3.23 must therefore hold for each of the individual basis functions resulting in N equations like 3.23, namely

$$(\psi_n(\mathbf{r}), V_0(\mathbf{r})) = (\psi_n(\mathbf{r}), L(V(\mathbf{r}))) \quad n = 1, ..., N. \tag{3.25}$$

The unknown potential is transformed to something more tractable for numerical computing by approximating it as a linear combination of N linearly independent basis functions $\varphi_n(\mathbf{r})$ $(n = 1, ..., N)$

$$V(\mathbf{r}) \simeq \sum_{n=1}^{N} v_n \varphi_n(\mathbf{r}) \tag{3.26}$$

where v_n are *nodal parameters* which are functions of the nodal points $\mathbf{r}_n$. The most commonly used basis functions are planar triangles with either a constant potential or linearly varying potential across the surface of each triangle. The unknown coefficients therefore control the approximations to the true potentials.

Naturally these approximations lead to errors that must be controlled using the basis functions. Common error control methods are *collocation* and *Galerkin* weighting. The simpler method of these two is collocation weighting, in it the error is controlled at the same discrete locations as the nodal points. In Galerkin weighting the error is controlled as either constant or linear function across the entire triangle.

This short introduction to BEM head modeling should indicate that in all cases the selection of weighting basis functions and potential basis functions lead to an $N \times N$ system of equations of the form $\mathbf{g} = \mathbf{H}\mathbf{v}$. The solution for $\mathbf{v}$ can then be expressed as

$$\mathbf{v} = \mathbf{H}^{-1}\mathbf{g}$$

where

$$\mathbf{g} = \mathbf{G}_0\mathbf{q} = \begin{bmatrix} (\psi_n(\mathbf{r}), k_0(\mathbf{r}, \mathbf{r}_q))^T \\ \vdots \\ (\psi_n(\mathbf{r}), k_0(\mathbf{r}, \mathbf{r}_q))^T \end{bmatrix} \mathbf{q}$$

with

$$k_0(\mathbf{r}, \mathbf{r}_q)) = \frac{1}{4\pi} \frac{\mathbf{r} - \mathbf{r}_q}{|\mathbf{r} - \mathbf{r}_q|^3}.$$

The potential, $V(\mathbf{r})$, can now be found from equation 3.26 using these coefficients

$$V(\mathbf{r}) \simeq [\varphi_1(\mathbf{r}), ..., \varphi_N(\mathbf{r})]\mathbf{v} = [\varphi_1(\mathbf{r}), ..., \varphi_N(\mathbf{r})]\mathbf{H}^{-1}\mathbf{G}_0\mathbf{q}. \tag{3.27}$$

The solution can be further partitioned into once-per-subject computation and run-time computation terms. With the dipoles restricted to the cortex surface and orientation equal to the surface normal the run-time terms are simply the dipole moments (also referred to as simply the sources in this text).

Although this is an improvement from the spherical head shapes the computational overhead is significant. The BEM method still also assumes homogeneity and isotropy within each region of the head, ignoring for example anisotropy in

Head volume	**Conductivity**, σ $[(\Omega m)^{-1}]$
Brain	0.33
Skull	0.0042
Scalp	0.33

Table 3.1: *Typical conductivity values used for different parts of the head volume. A detailed discussion on the topic of tissue conductivity can be seen in [13] pp. 151-158.*

the white matter and in the skull. Conductivity values typically used in the bioelectromagnetism community assume the skull to be 40-90 times more resistive than the brain and scalp [1]. The values used in this thesis for the conductivity of the brain, skull and scalp are listed in table 3.1.

3.5 BrainStorm

The discussion so far on the forward model indicates the complexity and importance of it for EEG research. In this thesis the main focus is on solving the inverse EEG problem but to do so a forward model is needed. This forward model is obtained using a Matlab software package called *BrainStorm* [25], free but copyrighted software available under the terms of the GNU General Public License . The package implements many different head modeling techniques which have been discussed above. In the simulations and tests presented later the head models are obtained using methods and functions implemented in BrainStorm. This section therefore describes the basic steps of creating the head model from tesselation surfaces obtained from MRI data and how a head model can be obtained using electrode channel location if no MRI data is available.

The first step in the head modeling process is acquiring the tesselation surfaces for different head volumes, which typically include brain, inner skull, outer skull and scalp, sometimes also CSF (e.g. see figure 3.3 on page 20). Anatomical images are necessary for this and commonly MR Images are used. The details of tesselation surface extraction from MRI data is beyond the scope of this text. *BrainSuite* [26] is a software package for surface extraction compatible with BrainStorm and available online for research purposes. The EEG electrode placement on the subjects head is very important. This requires electrode channel placements being aligned with the tesseleation surfaces. With surfaces and EEG channels aligned a forward model can be calculated. Remember that the forward model provides the information on how dipole moments $\mathbf{s}$ located inside the head produce measureable voltages $\mathbf{m}$ on the scalp. Here we furthermore use the cortical tesselation surface to restrict the sources to be placed vertically on its nodal points. Now using the head models introduced in last section equation 3.8 can be solved to give the forward model. With the algebraic formulation from section 3.3 and remembering that the dipole orientations are fixed the head modeling procedure gives us the lead field matrix **A** where in the noise free case the relationship between the sources $\mathbf{s} = [q_1, ..., q_P]^T$ and potentials (EEG measurements) **m** is

$$\mathbf{m} = \mathbf{A}\mathbf{s}.$$

The forward modeling is therefore necessary to give us the lead field matrix **A** of the system.

3.5.1 Phantom

In some cases there is no anatomical information available on the subjects head or simulated data testing is needed for validation purposes. *Collins et al.* [27] presented the design and creation of a realistic, digital, volumetric phantom of the human head. It is made up of ten volumetric data sets that define the spatial distribution for different tissues (e.g. gray matter, white matter, skin, etc.). This brain phantom, often referred to as the *Montreal Brain Phantom*, is intended for validation of medical image processing algorithms on simulated data since in most cases it is impossible to establish ground truth with *in vivo* data, as is the case for EEG. By using the brain phantom one can evaluate his algorithm or procedure in a controlled fashion. By a method called *thin-plate-spline (TPS)* warping *Darvas et al.* [28] fitted the Montreal Brain Phantom to a subject's head using the EEG electrode placement. This method is implemented in BrainStorm and used to generate tesselation surfaces from electrode placements when no MRI data (or tesselation data) is available. Since this method is used when creating a head model in some of the simulations and tests in chapters 5 and 6 we briefly go through the steps here.

Montreal Phantom Warping Using Channel Locations

3D coordinates of the subjects EEG channel locations are necessary. The first step is to align the coordinate systems of the Montreal Phantom and the EEG channels. Figure 3.5 shows 256 EEG channels as yellow dots and the inner skull tesselation surface of the Montreal Phantom. The illustration on the left shows how the coordinate systems are rotated with respect to each other. On the right illustration the coordinate systems have been aligned. After alignment it can be seen from the figure that the channels still do not match the Phantom

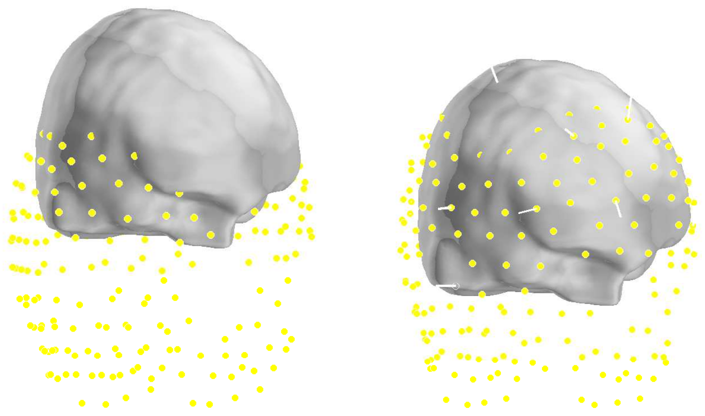

Figure 3.5: *The figure to the left shows that there is a* 90° *rotation between the EEG channels (yellow dots) and the Montreal Phantom coordinates. On the right the coordinate systems are aligned but some channels are placed inside of the surface (inner skull).*

head model since some channels are placed inside of the skull. On figure 3.6 the Phantom head model has been warped using TPS to match the channel locations. One must keep in mind that this is only a head model template. For artificial simulations this does not matter since all forward calculations are done using this head model. However for real data testing this head model is only a crude estimation of the subjects head and some errors inevitably will be generated because of that.

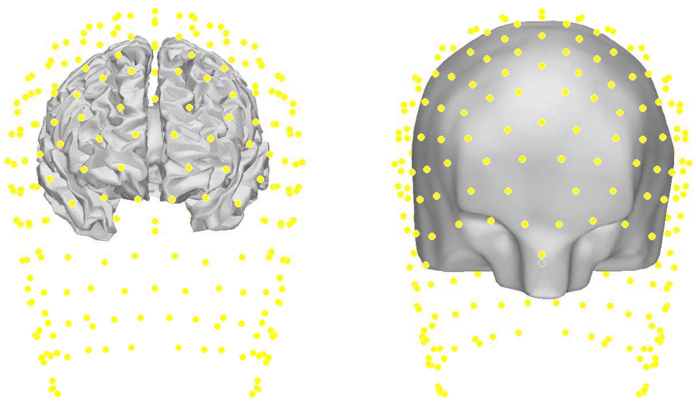

Figure 3.6: *After using TPS warping a Phantom head model has been created that matches the channel locations. The surfaces show are the brain and outer skull, on the left and right respectively.*

Chapter 4

The Linear Inverse Problem

With the forward model established in previous chapter the ill-posed inverse problem can be tackled. Using a lead field matrix $\mathbf{A}$ and a vector $\mathbf{m}$ of EEG measurements the highly underdetermined task is to estimate the vector of source dipole magnitudes $\mathbf{s}$. This will be done using Bayes' theorem. Constraints are therefore applied to the problem by specifying probability distributions for the noise and sources, called *likelihood* and *prior* respectively. The parameters describing these two distributions will be referred to as the *hyperparameters* in the text. By adjusting the hyperparameters using the data in $\mathbf{m}$ the *posterior* probability can be determined, which specifies the most probable value for the sources. The first approach will assume Gaussian distributions for both the noise and sources. Thus only two hyperparameters need to be adjusted, namely the variances of the two distributions. More complicated prior distributions will then be applied which incorporate sparsity and spatial smoothing into the source space. The number of hyperparameters increases drastically for these cases. The first part of this chapter is mostly based on the final chapter in a book by *Bishop* [7] and papers by *MacKay* [9, 10], where this method is used on Neural Networks. Second part of this chapter uses *Automatic relevance determination (ARD)* [8, 10, 11, 12]. Different iterative update algorithms for hyperparameter estimation will be presented and in the ARD algorithms there are some original update formulas the author has not found in the literature. In chapter 5 evaluation and testing of the algorithms will be presented.

Before starting the formulation lets look at a simple example which indicates the complexity of the problem at hand. As was noted in previous chapters we assume primary currents to be aligned normally to the cortical surface. So assigning a current dipole to each node of the triangular tessellation elements on the cortical surface means the dipole orientation is fixed to equal the local surface normal. Using the algebraic formulations from section 3.3 the inverse problem is linear and the only unknowns are the dipole amplitudes in each of the tessellation elements (i.e. the elements of $\mathbf{s}$). Now since $\mathbf{A}$ from equation 3.11 is fixed one could simply try to estimate the sources from the measurements $\mathbf{m}$ using the well known method of least squares, i.e. the calculated sources would be

$$\hat{\mathbf{s}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{m}.$$

But assuming the number of sensors N is in the order of 10^2 and the number of unknowns P is in the order of 10^4 the problem is severely underdetermined. Any measurement noise would be amplified beyond any acceptable level. Lets illustrate this with a simple example. We define a discrete gray scale source figure of dimensions 16×16 and reorder its elements in a vector $\mathbf{s}$ of length $P = 16 \times 16 = 256$. Then we warp it using a random matrix $\mathbf{A}$ with its elements drawn from a Gaussian distribution to generate our signals $\mathbf{m}$. Simulating an actual measurement some Gaussian noise is added, with standard deviation 0.01. The resulting signal-to-noise ratio is therefore around 44dB which is rather high. Lets see what happens if we now simply use the method of least squares to estimate the source image and how the dimensions of $\mathbf{A}$ influence the result. On figure 4.1 we can see the true source in the top left corner. Next to it we can see the least squares source estimate if $\mathbf{A}$ has dimensions 256×256. In the two figures below the dimensions of $\mathbf{A}$ have been reduced slightly in the sense that the number of "measured" values N is reduced. The source estimates in these cases is unacceptable even though the problem is far from being as underdetermined as the EEG inverse problem.

4.1 Bayesian Formulation

The forward model which will be used from now on is based on the algebraic formulation from equation 3.11. To account for some measurement noise an additive noise vector $\boldsymbol{\epsilon}$ is added to the model. The forward model will therefore be written

$$\mathbf{m} = \mathbf{A}\mathbf{s} + \boldsymbol{\epsilon} \tag{4.1}$$

where $\mathbf{m}$ is an $N \times 1$ vector of EEG signals, $\mathbf{A}$ is the $N \times P$ lead field matrix and $\mathbf{s}$ is a $P \times 1$ vector of source dipole magnitudes to be estimated. For now the noise vector is not specified further, that will be done when the likelihood

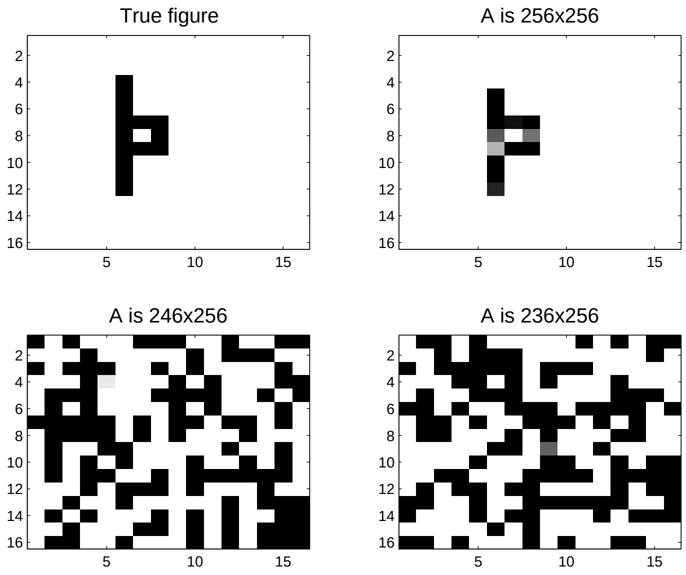

Figure 4.1: *Estimation of a source figure* $\mathbf{s}$ *from measurements* $\mathbf{m}$ *for different sizes* N *of* $\mathbf{m}$*. The method of least squares is used and the matrix* $\mathbf{A}$ *is generated by drawing numbers from a normal distribution. The true source is plotted in the top left corner. Three other figure estimates are plotted for different dimensions of* $\mathbf{A}$*. They clearly show how the estimate gets worse as the number of "measured" values* N *is reduced. The signal-to-noise ratio is very high, or around 44dB for all examples.*

is introduced below. The noise free signal $\mathbf{m}_0$ is then

$$\mathbf{m}_0 = \mathbf{A}\mathbf{s} \tag{4.2}$$

Using Bayes' theorem the posterior probability can be written

$$p(\mathbf{s}|\mathbf{m}) = \frac{p(\mathbf{m}|\mathbf{s})p(\mathbf{s})}{p(\mathbf{m})} \tag{4.3}$$

or writing the theorem in words

$$\textit{posterior} = \frac{\textit{likelihood} \times \textit{prior}}{\textit{normalizing const.}}. \tag{4.4}$$

In general the prior can be written as an exponential of the form

$$p(\mathbf{s}) = \frac{1}{Z_s(\alpha)} \exp(-\alpha E_s) \tag{4.5}$$

where

$$Z_s(\alpha) = \int \exp(-\alpha E_s) d\mathbf{s} \tag{4.6}$$

is a normalization factor ensuring that $\int p(\mathbf{s})d\mathbf{s} = 1$. For now it is assumed the hyperparameter α is known. Lets call E_s the decay function, it controls what type of prior distribution is assumed. The decay function relates directly to Tikhonov regularization and what type of regularization norm is used. Here

$$E_s = \frac{1}{2}|\mathbf{s}|^2 = \frac{1}{2}\sum_{i=1}^{P} s_i^2 \tag{4.7}$$

is chosen which gives a Gaussian prior distribution enabling us to proceed with rather straight forward analytical calculations. The prior then takes the form

$$p(\mathbf{s}) = \frac{1}{Z_s(\alpha)} \exp\left(-\frac{\alpha}{2}|\mathbf{s}|^2\right) \tag{4.8}$$

and using the identity $\int_{-\infty}^{\infty} \exp(\frac{\lambda}{2}x^2)dx = \left(\frac{2\pi}{\lambda}\right)^{1/2}$ gives

$$\begin{aligned} Z_s(\alpha) &= \int_{-\infty}^{\infty} \exp\left(-\frac{\alpha}{2}\sum_{i=1}^{P} s_i^2\right) ds_1 \cdots ds_P \\ &= \prod_{i=1}^{P} \int_{-\infty}^{\infty} \exp\left(-\frac{\alpha}{2}s_i^2\right) ds_i \\ &= \prod_{i=1}^{P} \left(\frac{2\pi}{\alpha}\right)^{1/2} = \left(\frac{2\pi}{\alpha}\right)^{P/2}. \end{aligned} \tag{4.9}$$

As for the prior the likelihood distribution can in general be written in exponential form

$$p(\mathbf{m}|\mathbf{s}) = \frac{1}{Z_m(\beta)} \exp(-\beta E_m) \tag{4.10}$$

with normalization factor

$$Z_m(\beta) = \int \exp(-\beta E_m) d\mathbf{m} \tag{4.11}$$

where β is another hyperparameter assumed to be known for now and E_m is an error function. Using the sum-of-squares error function given by

$$E_m = \frac{1}{2}|\mathbf{m} - \mathbf{A}\mathbf{s}|^2 \tag{4.12}$$

means that the noise on the measured data is assumed Gaussian with zero mean. Then as for the prior we get the likelihood

$$p(\mathbf{m}|\mathbf{s}) = \frac{1}{Z_m(\beta)} \exp\left(-\frac{\beta}{2}|\mathbf{m} - \mathbf{A}\mathbf{s}|^2\right). \tag{4.13}$$

Using the same methods calculating the normalization factor as we used for the prior gives

$$Z_m(\beta) = \prod_{n=1}^{N} \int_{-\infty}^{\infty} \exp\left(-\frac{\beta}{2}(m_n - \mathbf{a}_n^T\mathbf{s})^2\right) dm_n$$

where $\mathbf{a}_n^T$ is the n-th row vector of $\mathbf{A}$. Then using change of variables, i.e. $\xi_n = m_n - \mathbf{a}_n^T\mathbf{s}$ and then $d\xi_n = dm_n$, we get

$$\begin{aligned} Z_m(\beta) &= \prod_{n=1}^{N} \int_{-\infty}^{\infty} \exp\left(-\frac{\beta}{2}\xi_n^2\right) d\xi_n \\ &= \left(\frac{2\pi}{\beta}\right)^{N/2}. \end{aligned} \tag{4.14}$$

Having defined identities for both the prior and the likelihood Bayes' theorem can be used to give an equation for the posterior, i.e.

$$p(\mathbf{s}|\mathbf{m}) = \frac{1}{Z_L(\alpha,\beta)} \exp(-L(\mathbf{s},\alpha,\beta)) \tag{4.15}$$

where

$$Z_L(\alpha,\beta) = \int \exp(-L(\mathbf{s},\alpha,\beta))d\mathbf{s} \tag{4.16}$$

and

$$L(\mathbf{s},\alpha,\beta) = \beta E_m + \alpha E_s. \tag{4.17}$$

Note that $L(\mathbf{s},\alpha,\beta)$ and $L(\mathbf{s})$ will be used interchangeably. Before going further lets summarize a bit the above. The prior and likelihood were assumed Gaussian enabling us to calculate the posterior using Bayes' theorem. Two hyperparameters were introduced, one for the prior and the other for the likelihood. For now we assume these parameters are known but later it will be shown how one can estimate them.

Moving on we consider the important part of finding the most probable value of $\mathbf{s}$ by finding the maximum of the posterior distribution. Instead of maximizing the posterior distribution directly it is more convenient to maximize the logarithm of the posterior. This is equivalent since the logarithm is a monotonic increasing function. Lets call this most probable vector $\mathbf{s}_{MP}$. Since $Z_L(\alpha,\beta)$ is independent of $\mathbf{s}$ then maximizing $\ln(p(\mathbf{s}|\mathbf{m}))$ with respect to $\mathbf{s}$ is equivalent to maximizing

$$-L(\mathbf{s},\alpha,\beta) = -\beta E_m - \alpha E_s = -\frac{\beta}{2}|\mathbf{m} - \mathbf{A}\mathbf{s}|^2 - \frac{\alpha}{2}|\mathbf{s}|^2.$$

Differentiation gives

$$\begin{aligned} -\frac{\partial}{\partial \mathbf{s}} L(\mathbf{s}) &= \frac{\partial}{\partial \mathbf{s}}\left(\frac{\beta}{2}|\mathbf{m} - \mathbf{A}\mathbf{s}|^2\right) + \frac{\partial}{\partial \mathbf{s}}\left(\frac{\alpha}{2}|\mathbf{s}|^2\right) \\ &= \beta(\mathbf{A}^T\mathbf{m} - \mathbf{A}^T\mathbf{A}\mathbf{s}) - \alpha\mathbf{s} \end{aligned} \tag{4.18}$$

resulting in the most probable value

$$\mathbf{s}_{MP} = \left(\mathbf{A}^T\mathbf{A} + \frac{\alpha}{\beta}\mathbf{I}\right)^{-1}\mathbf{A}^T\mathbf{m}. \tag{4.19}$$

The differentiation above is straight forward but tedious, it is therefore listed in more detail in appendix A.3.1. Using an identity presented later in the text (see equation 4.66) the most probable value can equivalently be written

$$\mathbf{s}_{MP} = \mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T + \frac{\alpha}{\beta}\mathbf{I}\right)^{-1}\mathbf{m}.$$

The equations so far are exact although $Z_L(\alpha,\beta)$ cannot be calculated analytically. Furthermore the calculation of $Z_L(\alpha,\beta)$ is over the space of $\mathbf{s}$ which in practice is very large. A simplifying expression is therefore introduced. *MacKay* [10] was dealing with neural networks and introduced a Gaussian approximation for the posterior distribution. He obtained it by considering the Taylor expansion around its minimum value and retaining terms up to the second order. It turns out that doing the same here is equivalent to completing the square over $\mathbf{s}$ (e.g. see page 167 in [8]), except that here the identity is exact. Considering the expansion of $L(\mathbf{s},\alpha,\beta)$ around its minimum value gives

$$L(\mathbf{s}) = L(\mathbf{s}_{MP}) + \frac{1}{2}(\mathbf{s}-\mathbf{s}_{MP})^T\mathbf{H}(\mathbf{s}-\mathbf{s}_{MP}) \tag{4.20}$$

where $\mathbf{H}$ is the $P \times P$ Hessian matrix of the total error function, i.e. defining ∇ as the gradient in the space of $\mathbf{s}$ the Hessian is

$$\begin{aligned}\mathbf{H} &= \nabla\nabla L(\mathbf{s},\alpha,\beta)\\ &= \beta\nabla\nabla E_m + \alpha\nabla\nabla E_s\\ &= \beta\mathbf{A}^T\mathbf{A} + \alpha\mathbf{I}.\end{aligned} \tag{4.21}$$

More detailed calculations of the Hessian are shown in appendix A.3.1. This expansion now leads to a Gaussian posterior distributions around $\mathbf{s}_{MP}$ given by

$$p(\mathbf{s}|\mathbf{m}) = \frac{1}{Z_L^*}\exp\left(-L(\mathbf{s}_{MP}) - \frac{1}{2}(\mathbf{s}-\mathbf{s}_{MP})^T\mathbf{H}(\mathbf{s}-\mathbf{s}_{MP})\right) \tag{4.22}$$

where $Z_L^* = Z_L$ is a normalization constant appropriate to the expansion, given by

$$Z_L^* = \int \exp\left(-L(\mathbf{s}_{MP}) - \frac{1}{2}(\mathbf{s}-\mathbf{s}_{MP})^T\mathbf{H}(\mathbf{s}-\mathbf{s}_{MP})\right)d\mathbf{s}.$$

With the posterior distribution rewritten to this form allows a great deal of progress to be made analytically.

Evaluation of the normalization constant Z_L^* is straight forward but involves some lengthy calculations. The most important steps will be listed here [1]. It is convenient to work in terms of the eigenvectors $\mathbf{u}_k$ and eigenvalues λ_k of $\mathbf{H}$ $(k = 1, ..., P)$. So lets consider the eigenvector equation for $\mathbf{H}$

$$\mathbf{H}\mathbf{u}_k = \lambda_k \mathbf{u}_k.$$

The vector $\mathbf{s} - \mathbf{s}_{MP}$ can be expanded as a linear combination of the eigenvectors

$$\mathbf{s} - \mathbf{s}_{MP} = \sum_{k=1}^{P} \xi_k \mathbf{u}_k.$$

By change of variables the integration can be done over $d\xi_1 ... d\xi_P$ instead of $ds_1 ... ds_P$. Furthermore using the orthonormality of $\mathbf{u}_k$ the normalization constant becomes

$$Z_L^* = \prod_{k=1}^{P} \int_{-\infty}^{\infty} \exp\left(-L(\mathbf{s}_{MP}) - \frac{\lambda_k \xi_k^2}{2}\right) d\xi_k$$

which can be simplified to

$$Z_L^* = \exp(-L(\mathbf{s}_{MP}))(2\pi)^{P/2} \det(\mathbf{H})^{-1/2} \tag{4.23}$$

4.1.1 Framework for Hyperparameters α and β

Until now the values of α and β have been assumed to be known but. In general the values are unknown and have to be estimated from the data. Treatment of hyperparameters involves *Occam's razor* [2] since the hyperparameters which give the best fit to the data represent over fitting and do not give the best generalization. The correct treatment for unknown parameters is to integrate them out of the predictions. For example the posterior distribution of dipole amplitudes is given by

$$\begin{aligned} p(\mathbf{s}|\mathbf{m}) &= \iint p(\mathbf{s}, \alpha, \beta|\mathbf{m}) d\alpha d\beta \\ &= \iint p(\mathbf{s}|\alpha, \beta, \mathbf{m}) p(\alpha, \beta|\mathbf{m}) d\alpha d\beta. \end{aligned} \tag{4.24}$$

The approach used here is called the *evidence approximation*, e.g. discussed by MacKay [9].

[1]for more details see appendix B in Bishop [7].

[2]Named after William of Occam (1285-1349), and is the principle that simple models should be preferred over more complex ones.

Assuming the posterior probability distribution $p(\alpha, \beta|\mathbf{m})$ is sharply peaked around the most probable values of the hyperparameters α_{MP} and β_{MP} then we can write

$$\begin{aligned} p(\mathbf{s}|\mathbf{m}) &\simeq p(\mathbf{s}|\alpha_{MP}, \beta_{MP}, \mathbf{m}) \iint p(\alpha, \beta|\mathbf{m}) d\alpha d\beta. &(4.25)\\ &= p(\mathbf{s}|\alpha_{MP}, \beta_{MP}, \mathbf{m}) &(4.26)\end{aligned}$$

What this says is that we should find the hyperparameters which maximize the posterior probability and use them for remaining calculations. To find α and β we need to evaluate the posterior given by

$$p(\alpha, \beta|\mathbf{m}) = \frac{p(\mathbf{m}|\alpha, \beta)p(\alpha, \beta)}{p(\mathbf{m})} \tag{4.27}$$

which requires a choice for the prior $p(\alpha, \beta)$, sometimes called the hyperprior. Since we have no idea of what are suitable values for α and β the prior should be chosen so that it gives in some sense equal weight to all possible values. Thus we assume the prior is very flat, such priors are often called *non-informative*. By choosing a very flat prior means it is very insensitive to the values of α and β. The denominator in 4.27 is independent of α and β and the prior $p(\alpha, \beta)$ is assumed very flat. Maximizing the posterior $p(\alpha, \beta|\mathbf{m})$ can therefore be done by maximizing $p(\mathbf{m}|\alpha, \beta)$, called the *evidence* for α and β. Our overall Bayesian analysis is proceeding in a hierarchical fashion. First level is determining the distribution of $\mathbf{s}$ and now we are seeking the distribution for the hyperparameters. Writing in terms of parameters already established in equations 4.5, 4.9, 4.10 and 4.14 we get

$$\begin{aligned} p(\mathbf{m}|\alpha, \beta) &= \int p(\mathbf{m}|\mathbf{s}, \alpha, \beta)p(\mathbf{s}|\alpha, \beta)d\mathbf{s} \\ &= \int p(\mathbf{m}|\mathbf{s}, \beta)p(\mathbf{s}|\alpha)d\mathbf{s} \\ &= \frac{1}{Z_m(\beta)Z_s(\alpha)} \int \exp(-L(\mathbf{s}, \alpha, \beta))d\mathbf{s} \\ &= \left(\frac{2\pi}{\beta}\right)^{-N/2} \left(\frac{2\pi}{\alpha}\right)^{-P/2} Z_L(\alpha, \beta). &(4.28)\end{aligned}$$

Using $Z_L^* = Z_L$ from equation 4.23 gives

$$p(\mathbf{m}|\alpha, \beta) = \left(\frac{2\pi}{\beta}\right)^{-N/2} \left(\frac{2\pi}{\alpha}\right)^{-P/2} \exp(-L(\mathbf{s}_{MP}))(2\pi)^{P/2} \det(\mathbf{H})^{-1/2}. \tag{4.29}$$

The natural logarithm of the evidence is then

$$\begin{aligned}\ln p(\mathbf{m}|\alpha,\beta) &= -\frac{N}{2}\ln\frac{2\pi}{\beta} - \frac{P}{2}\ln\frac{2\pi}{\alpha} - L(\mathbf{s}_{MP}) + \frac{P}{2}\ln 2\pi - \frac{1}{2}\ln\det(\mathbf{H}) \\ &= -L(\mathbf{s}_{MP}) - \frac{1}{2}\ln\det(\mathbf{H}) \\ &\quad +\frac{P}{2}\ln\alpha + \frac{N}{2}\ln\beta - \frac{N}{2}\ln 2\pi. \qquad (4.30)\end{aligned}$$

Lets first find the maximum with respect to α. Equation 4.21 gives the Hessian:

$$\mathbf{H} = \beta\mathbf{A}^T\mathbf{A} + \alpha\mathbf{I}.$$

Let λ_i $(i = 1, ..., P)$ be the eigenvalues of $\beta\mathbf{A}^T\mathbf{A}$, then $\mathbf{H}$ has eigenvalues $\lambda_i + \alpha$ and

$$\begin{aligned}\frac{\partial}{\partial\alpha}\ln\det(\mathbf{H}) &= \frac{\partial}{\partial\alpha}\ln\left(\prod_{i=1}^{P}(\lambda_i+\alpha)\right) = \frac{\partial}{\partial\alpha}\sum_{i=1}^{P}\ln(\lambda_i+\alpha) \\ &= \sum_{i=1}^{P}\frac{1}{\lambda_i+\alpha} = \mathrm{Tr}(\mathbf{H}^{-1}).\end{aligned}$$

Using this along with $L(\mathbf{s}_{MP}) = \beta E_m(\mathbf{s}_{MP}) + \alpha E_s(\mathbf{s}_{MP})$ the differentiation of the log-evidence is straight forward

$$\begin{aligned}\frac{\partial}{\partial\alpha}\ln p(\mathbf{m}|\alpha,\beta) &= -E_s(\mathbf{s}_{MP}) - \frac{1}{2}\mathrm{Tr}(\mathbf{H}^{-1}) + \frac{P}{2\alpha} \\ &= -E_s(\mathbf{s}_{MP}) - \frac{1}{2}\sum_{i=1}^{P}\frac{1}{\lambda_i+\alpha} + \frac{P}{2\alpha}.\end{aligned}$$

Setting this derivative to zero to locate the maximum gives

$$\begin{aligned}2\alpha E_s(\mathbf{s}_{MP}) &= P - \sum_{i=1}^{P}\frac{\alpha}{\lambda_i+\alpha} = \sum_{i=1}^{P}1 - \sum_{i=1}^{P}\frac{\alpha}{\lambda_i+\alpha} = \sum_{i=1}^{P}\frac{\lambda_i+\alpha-\alpha}{\lambda_i+\alpha} \\ &= \sum_{i=1}^{P}\frac{\lambda_i}{\lambda_i+\alpha} \equiv \gamma. \qquad (4.31)\end{aligned}$$

Now we have an equation which specifies the value of α which maximizes the evidence. Next step is to do the same for β. Since λ_i are the eigenvalues of $\beta\mathbf{A}^T\mathbf{A}$ we see that λ_i is directly proportional to β and hence

$$\frac{\partial\lambda_i}{\partial\beta} = \frac{\lambda_i}{\beta}$$

then

$$\begin{aligned}\frac{\partial}{\partial\beta}\ln\det(\mathbf{H}) &= \frac{\partial}{\partial\beta}\sum_{i=1}^{P}\ln(\lambda_i+\alpha) = \sum_{i=1}^{P}\left(\frac{1}{\lambda_i+\alpha}\frac{\partial\lambda_i}{\partial\beta}\right)\\ &= \frac{1}{\beta}\sum_{i=1}^{P}\frac{\lambda_i}{\lambda_i+\alpha}.\end{aligned}$$

And as we did before we differentiate the log-evidence but now with respect to β

$$\frac{\partial}{\partial\beta}\ln p(\mathbf{m}|\alpha,\beta) = -E_m(\mathbf{s}_{MP}) - \frac{1}{2\beta}\sum_{i=1}^{P}\frac{\lambda_i}{\lambda_i+\alpha} + \frac{N}{2\beta}$$

setting this to zero gives then an equation for β at the maximum

$$2\beta E_m(\mathbf{s}_{MP}) = N - \sum_{i=1}^{P}\frac{\lambda_i}{\lambda_i+\alpha} = N - \gamma. \tag{4.32}$$

Now we have two criteria for finding the optimum values for the hyperparameters α and β. In practical applications of the approach described so far one needs to find the optimum hyperparameters as well as the optimum source vector $\mathbf{s}_{MP}$. A simple approach to solve this problem is to use an iterative algorithm. Finally it is worth noting that an alternative to the evidence approximation used here a technique called the *Expectation Maximization (EM)* algorithm can be used which converges to the same solution, as shown in [8] (pp. 448-450).

4.1.2 Algorithm I for Parameter Estimation

Here a simple iterative algorithm will be presented that uses the equations already derived to optimize the hyperparameters α and β while also estimating the optimum source vector $\mathbf{s}_{MP}$.

Algorithm I

(i) Initialize hyperparameters α and β as 1.

(ii) Estimate $\mathbf{s}_{MP}$ using equation 4.19, i.e.

$$\mathbf{s}_{MP} = \left(\mathbf{A}^T\mathbf{A} + \frac{\alpha}{\beta}\mathbf{I}\right)^{-1}\mathbf{A}^T\mathbf{m}.$$

(iii) Re-estimate α and β using the criteria from equations 4.31 and 4.32:

$$\begin{aligned}\alpha_{new} &= \frac{\gamma}{2E_s(\mathbf{s}_{MP})} \\ \beta_{new} &= \frac{N-\gamma}{2E_m(\mathbf{s}_{MP})}\end{aligned}$$

where $\gamma = \sum_{i=1}^{P} \frac{\lambda_i}{\lambda_i+\alpha}$ and λ_i are the eigenvalues of $\beta\mathbf{A}^T\mathbf{A}$.

(iv) If convergence criterion fulfilled then stop, else go back to step (ii).

The convergence of the algorithm is evaluated using the change in α and β between iterations. If the larger change of the two is smaller than some predetermined threshold value $\varepsilon > 0$ the algorithm is stopped. Or in more mathematical way the algorithm stops at iteration i if $K(i) < \varepsilon$ where

$$K(i) = \max\{|\alpha(i) - \alpha(i-1)|, |\beta(i) - \beta(i-1)|\}\,. \tag{4.33}$$

Regarding the initialization of hyperparameters the author also tried using random numbers between 0 and 1 drawn from a uniform distribution. In general the difference between the two methods was not great. The benefit of initializing with constant numbers is reproducibility of results.

4.1.3 Performance Evaluation

Now lets look at equations 4.8 and 4.13 for the prior and likelihood distributions. For our choice of E_m and E_s both of these distributions are Gaussian so if the algorithm converges the inverse alpha and beta values should equal the variances of the prior and likelihood respectively, i.e.

$$\sigma^2_{p(s)} = \frac{1}{\alpha} \tag{4.34}$$

$$\sigma^2_{p(m|s)} = \frac{1}{\beta}. \tag{4.35}$$

In chapter 5 these equations will be used to test the algorithm, e.g. using artificial data where the variances are known we can see whether the algorithm returns the correct values. For real data where one has little or no knowledge of the variances the algorithm should therefore give us estimates of the signal and noise variances. Naturally these estimates should be positive since they represent variance values.

Lets therefore check if the update equations are guaranteed to give positive outcomes. First step is to look at the eigenvalues of $\beta\mathbf{A}^T\mathbf{A}$ (or equivalently $\beta\mathbf{A}\mathbf{A}^T$), this matrix is real symmetric. Eigenvectors of real symmetric matrices can be chosen to be orthonormal. More importantly this matrix is positive semidefinite meaning that all eigenvalues are non-negative, i.e. $\lambda_i \geq 0$ for all $i = 1, ..., P$. Since all eigenvalues are non-negative and initializing α with a positive number then $\gamma \geq 0$. Furthermore we know that $E_s(\mathbf{s}_{MP})$ is always positive by definition so we have shown that

$$\alpha_{new} \geq 0 \tag{4.36}$$

as long as the initialization value is positive.

Next lets check the update equation for β. By definition we know that $E_m(\mathbf{s}_{MP})$ is positive, so to demonstrate that β is non-negative we need to show that $N-\gamma$ is non-negative. From above we know that $\gamma \geq 0$, furthermore we know that we have N eigenvalues $\lambda_i \geq 0$ giving

$$\begin{aligned} N - \gamma &= N - \sum_{i=1}^{P} \frac{\lambda_i}{\lambda_i + \alpha} \\ &= N - \sum_{i=1}^{N} \frac{\lambda_i}{\lambda_i + \alpha} \end{aligned}$$

where $0 \leq \frac{\lambda_i}{\lambda_i+\alpha} \leq 1$ for all non-zero λ_i. This gives

$$\begin{array}{rc} & 0 \leq \sum_{i=1}^{N} \frac{\lambda_i}{\lambda_i+\alpha} \leq N \\ \Leftrightarrow & 0 \geq -\sum_{i=1}^{N} \frac{\lambda_i}{\lambda_i+\alpha} \geq -N \\ \Leftrightarrow & N \geq N - \sum_{i=1}^{N} \frac{\lambda_i}{\lambda_i+\alpha} \geq -N + N \\ \Leftrightarrow & N \geq N - \gamma \geq 0 \end{array}$$

which shows that $N - \gamma$ is non-negative and then

$$\beta_{new} \geq 0. \tag{4.37}$$

It can therefore be concluded that this algorithm is guaranteed to return non-negative α and β values. And for reasonable lead-field matrices where at least one eigenvalue is not zero then α and β are positive.

4.1.4 Algorithm Improvements

The algorithm just described is simple and implementing it on a computer is straight forward. But for our head models the $N \times P$ gain matrix $\mathbf{A}$ is usually very large where typical values for the dimensions could be $N = 256$ and

$P = 10000$. The matrix calculations in the algorithm therefore quickly become very memory and CPU intensive. Furthermore $\mathbf{A}$ is generally very ill-conditioned making the calculations more sensitive to numerical errors when doing finite resolution calculations on a computer. There are however some tricks that can be used to reduce memory consumption, speed up calculations and avoid build up of numerical errors.

Factorization called *singular value decomposition* (SVD) enables $N \times P$ matrix $\mathbf{A}$ to be written

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{4.38}$$

where $\mathbf{U}$ is $N \times N$ orthogonal [3] matrix of the eigenvectors of $\mathbf{A}\mathbf{A}^T$, $\mathbf{V}$ is $P \times P$ orthogonal matrix of the eigenvectors of $\mathbf{A}^T\mathbf{A}$ and $\mathbf{D}$ is $N \times P$ matrix with the square root of the eigenvalues of $\mathbf{A}\mathbf{A}^T$ on the diagonal, or simply the singular values, and zeros off the diagonal. SVD can be used to simplify the calculations in step (ii) of the algorithm. Using SVD on the equation in step (ii) gives

$$\begin{aligned}
\mathbf{s}_{MP} &= \left(\mathbf{A}^T\mathbf{A} + \frac{\alpha}{\beta}\mathbf{I}\right)^{-1} \mathbf{A}^T\mathbf{m} \\
&= \left((\mathbf{U}\mathbf{D}\mathbf{V}^T)^T\mathbf{U}\mathbf{D}\mathbf{V}^T + \frac{\alpha}{\beta}\mathbf{I}\right)^{-1} (\mathbf{U}\mathbf{D}\mathbf{V}^T)^T\mathbf{m} \\
&= \left(\mathbf{V}\mathbf{D}^T\mathbf{U}^T\mathbf{U}\mathbf{D}\mathbf{V}^T + \frac{\alpha}{\beta}\mathbf{I}\right)^{-1} \mathbf{V}\mathbf{D}^T\mathbf{U}^T\mathbf{m} \\
&= \left(\mathbf{V}\mathbf{D}^T\mathbf{D}\mathbf{V}^T + \frac{\alpha}{\beta}\mathbf{V}\mathbf{V}^T\right)^{-1} \mathbf{V}\mathbf{D}^T\mathbf{U}^T\mathbf{m} \\
&= \left(\mathbf{V}(\mathbf{D}^T\mathbf{D} + \frac{\alpha}{\beta}\mathbf{I})\mathbf{V}^T\right)^{-1} \mathbf{V}\mathbf{D}^T\mathbf{U}^T\mathbf{m} \\
&= \mathbf{V}\left(\mathbf{D}^T\mathbf{D} + \frac{\alpha}{\beta}\mathbf{I}\right)^{-1} \mathbf{V}^T\mathbf{V}\mathbf{D}^T\mathbf{U}^T\mathbf{m} \\
&= \mathbf{V}\left(\mathbf{D}^T\mathbf{D} + \frac{\alpha}{\beta}\mathbf{I}\right)^{-1} \mathbf{D}^T\mathbf{U}^T\mathbf{m}.
\end{aligned} \tag{4.39}$$

The $N \times N$ matrix $(\mathbf{D}^T\mathbf{D} + \frac{\alpha}{\beta}\mathbf{I})$ is diagonal simplifying the inverse calculations significantly since the inverse of a diagonal matrix is obtained simply by taking the scalar inverse of all the diagonal elements.

In step (iii) of the algorithm the eigenvalues of $\beta\mathbf{A}^T\mathbf{A}$ are calculated. This is a $P \times P$ matrix and since generally $N << P$ it is convenient to use the relationship

$$\text{eig}(\beta\mathbf{A}^T\mathbf{A}) = \beta \cdot \text{eig}(\mathbf{A}\mathbf{A}^T)$$

[3] $\mathbf{V}^{-1} = \mathbf{V}^T$ if the matrix is orthogonal.

that way the eigenvalues of the $N \times N$ matrix $\mathbf{A}\mathbf{A}^T$ are only calculated once and simply scaled after each iteration as β is updated.

It is worth noting that when the algorithm is run on systems with large $\mathbf{A}$ it is sometimes necessary to scale the data to avoid numerical problems caused by finite resolution in computers. For clarity the system equation is rewritten

$$\mathbf{m} = \mathbf{A}\mathbf{s} + \boldsymbol{\epsilon}$$

and scaling it with constant $K > 0$ gives

$$K\mathbf{m} = K\mathbf{A}\mathbf{s} + K\boldsymbol{\epsilon} \tag{4.40}$$

or equivalently

$$\mathbf{m}_s = \mathbf{A}\mathbf{s}_s + \boldsymbol{\epsilon}_s \tag{4.41}$$

where $\mathbf{m}_s = K\mathbf{m}$, $\mathbf{s}_s = K\mathbf{s}$ and $\boldsymbol{\epsilon}_s = K\boldsymbol{\epsilon}$ are the scaled sources, measurements and noise respectively. After running the algorithm on the scaled data the returned values therefore have to be scaled back to the original units.

4.2 Automatic Relevance Determination (ARD)

In previous section a single hyperparameter α was defined for the prior distribution and with the chosen decay function E_s the prior was Gaussian over the whole source space. In chapter 5 it will be shown how this can give good results for some examples. In the most general case one might expect the source distribution at a single time instance to be Gaussian distributed over the cortex, but for certain examples this is not the case. For example when using averaged evoked potentials (EPs), which were introduced in chapter 2.2, all background activity is averaged out and one would expect the sources to be localized and sparse. For this kind of non-Gaussian sources the Gaussian prior assumption is not a good one and we need to generalize the model from last section a bit more. *Automatic Relevance Determination* (ARD) [10, 11, 12] is a method that classifies the unknowns of the model. Using ARD here the sources s_k $(k = 1, ..., P)$ of $\mathbf{s}$ are divided into different classes and each class is assigned a hyperparameter. Effectively the cortex is therefore divided into classes. The ARD hyperparameters define the relevance of each class in the sense that if the sources belonging to the class are inactive their hyperparameter is high, vice versa if the sources are active the hyperparameter is low. In this way regions of inactive cortex area can then be switched of using the appropriate hyperparameters. That way the available data in $\mathbf{m}$ can be used on a smaller subspace of the source space. For the inverse EEG problem this is precisely what we want since the measurement space is much smaller than the source space.

For C different classes the ARD prior can be written as

$$p(\mathbf{s}) = \frac{1}{\prod_c Z_c} \exp\left(-\sum_{c=1}^{C} \alpha_c E_c\right) \tag{4.42}$$

where α_c are the hyperparameters for each class, E_c is the decay functions for the classes and $Z_c = \int \exp(-\alpha_c E_c) d\mathbf{s}$ are the normalization factors. Here we will use the same decay function as in last section for all classes and a single hyperparameter will be assigned to each source, meaning that the total number of classes will equal the number of sources ($C = P$). The prior can then be written

$$p(\mathbf{s}) = \frac{1}{Z_s(\alpha_1, ..., \alpha_P)} \exp\left(-\frac{1}{2}\sum_{i=1}^{P} \alpha_i s_i^2\right). \tag{4.43}$$

For further algebraic calculations it is convenient to write this in matrix notation by defining a diagonal matrix of α-values as

$$\mathbf{\Lambda} = \begin{bmatrix} \alpha_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \alpha_P \end{bmatrix}. \tag{4.44}$$

Then the prior can equivalently be written

$$p(\mathbf{s}) = \frac{1}{Z_s(\mathbf{\Lambda})} \exp\left(-\frac{1}{2}\mathbf{s}^T \mathbf{\Lambda} \mathbf{s}\right) \tag{4.45}$$

where the normalization factor is calculated in the same way as before

$$\begin{aligned} Z_s(\mathbf{\Lambda}) &= \int_{-\infty}^{\infty} \exp\left(-\frac{1}{2}\sum_{i=1}^{P} \alpha_i s_i^2\right) ds_1 ... ds_P \\ &= \prod_{i=1}^{P} \int_{-\infty}^{\infty} \exp\left(-\frac{1}{2}\alpha_i s_i^2\right) ds_i \\ &= \prod_{i=1}^{P} \left(\frac{2\pi}{\alpha_i}\right)^{1/2} = (2\pi)^{P/2} \prod_{i=1}^{P} \alpha_i^{-1/2}. \end{aligned} \tag{4.46}$$

In previous section the likelihood was Gaussian with single hyperparameter β, which will also be used here. Note that this means that the measurement noise is assumed to be Gaussian and the same over all electrodes. For clarity lets rewrite the equations (4.10 and 4.14) for the likelihood distribution from last section

$$p(\mathbf{m}|\mathbf{s}) = \frac{1}{Z_m(\beta)} \exp\left(-\frac{\beta}{2}|\mathbf{m} - \mathbf{A}\mathbf{s}|^2\right)$$

where

$$Z_m(\beta) = \left(\frac{2\pi}{\beta}\right)^{N/2}.$$

Before continuing further it is worth noting that the formulation here will be very similar to the one from last section. The only major difference is the prior distribution leading to more tedious calculations in the ARD case. Since the procedure here is so similar to the one from last section the derivation will not be as detailed. The reader should be able to go quickly through this section and if more details are required for better understanding one can always look at previous section.

Now Bayes theorem can be used to give an identity for the posterior (same as equation 4.15 from last section)

$$p(\mathbf{s}|\mathbf{m}) = \frac{1}{Z_L(\mathbf{\Lambda}, \beta)} \exp(-L(\mathbf{s}, \mathbf{\Lambda}, \beta))$$

with

$$Z_L(\mathbf{\Lambda}, \beta) = \int \exp(-L(\mathbf{s}, \mathbf{\Lambda}, \beta)) d\mathbf{s}$$

where

$$L(\mathbf{s}, \mathbf{\Lambda}, \beta) = \frac{\beta}{2}|\mathbf{m} - \mathbf{A}\mathbf{s}|^2 + \frac{1}{2}\mathbf{s}^T \mathbf{\Lambda} \mathbf{s}. \tag{4.47}$$

Maximizing the log-posterior is equivalent to maximizing $-L(\mathbf{s}, \mathbf{\Lambda}, \beta)$. Using the results from appendix A.3.2 gives

$$\frac{\partial}{\partial \mathbf{s}}(-L(\mathbf{s})) = \beta(\mathbf{A}^T\mathbf{m} - \mathbf{A}^T\mathbf{A}\mathbf{s}) - \mathbf{\Lambda}\mathbf{s} \tag{4.48}$$

and setting this equal to zero and solving for $\mathbf{s}$ gives the most probable value

$$\mathbf{s}_{MP} = \left(\mathbf{A}^T\mathbf{A} + \frac{1}{\beta}\mathbf{\Lambda}\right)^{-1} \mathbf{A}^T\mathbf{m}. \tag{4.49}$$

The similarity to equation 4.19 is obvious and as before the equations so far are exact analytical expressions. Using the same method as in last section an identity for $L(\mathbf{s})$ around the most probable value is considered (as in equation 4.20), written here again for clarity

$$L(\mathbf{s}) = L(\mathbf{s}_{MP}) + \frac{1}{2}(\mathbf{s} - \mathbf{s}_{MP})^T \mathbf{H} (\mathbf{s} - \mathbf{s}_{MP}).$$

The Hessian matrix $\mathbf{H}$ is then

$$\begin{aligned} \mathbf{H} &= \nabla\nabla L(\mathbf{s}, \mathbf{\Lambda}, \beta) \\ &= \beta\mathbf{A}^T\mathbf{A} + \nabla\nabla\left(-\frac{1}{2}\sum_{i=1}^{P} \alpha_i s_i^2\right) \\ &= \beta\mathbf{A}^T\mathbf{A} + \mathbf{\Lambda}. \end{aligned} \tag{4.50}$$

The posterior is then given by equation 4.22 and the normalization factor by equation 4.23, for clarity these equations are rewritten here:

$$\begin{aligned} p(\mathbf{s}|\mathbf{m}) &= \frac{1}{Z_L^*}\exp\left(-L(\mathbf{s}_{MP}) - \frac{1}{2}(\mathbf{s}-\mathbf{s}_{MP})^T\mathbf{H}(\mathbf{s}-\mathbf{s}_{MP})\right) \\ Z_L^* &= \exp(-L(\mathbf{s}_{MP}))(2\pi)^{P/2}\det(\mathbf{H})^{-1/2} \end{aligned}$$

where $Z_L^* = Z_L$.

4.2.1 Framework for Hyperparameters $\boldsymbol{\Lambda}$ and β

Now the hyperparameters need to be estimated with the best generalization in mind as was done in section 4.1.1. To find the most probable values for $\boldsymbol{\Lambda}$ and β we evaluate

$$p(\boldsymbol{\Lambda},\beta|\mathbf{m}) = \frac{p(\mathbf{m}|\boldsymbol{\Lambda},\beta)p(\boldsymbol{\Lambda},\beta)}{p(\mathbf{m})} \tag{4.51}$$

assuming a flat non-informative hyperprior $p(\boldsymbol{\Lambda},\beta)$. That way maximizing $p(\boldsymbol{\Lambda},\beta|\mathbf{m})$ can be done by maximizing the evidence $p(\mathbf{m}|\boldsymbol{\Lambda},\beta)$. Using equations 4.10, 4.14, 4.45 and 4.46 along with $Z_L^* = Z_L$ gives

$$\begin{aligned} p(\mathbf{m}|\boldsymbol{\Lambda},\beta) &= \int p(\mathbf{m}|\mathbf{s},\boldsymbol{\Lambda},\beta)p(\mathbf{s}|\boldsymbol{\Lambda},\beta)d\mathbf{s} \\ &= \int p(\mathbf{m}|\mathbf{s},\beta)p(\mathbf{s}|\boldsymbol{\Lambda})d\mathbf{s} \\ &= \frac{1}{Z_m(\beta)Z_s(\boldsymbol{\Lambda})}\int \exp(-L(\mathbf{s},\boldsymbol{\Lambda},\beta))d\mathbf{s} \\ &= \left(\frac{2\pi}{\beta}\right)^{-N/2}(2\pi)^{-P/2}\prod_{i=1}^{P}\alpha_i^{1/2}Z_L(\boldsymbol{\Lambda},\beta) \\ &= \left(\frac{2\pi}{\beta}\right)^{-N/2}(2\pi)^{-P/2}\prod_{i=1}^{P}\alpha_i^{1/2} \\ &\quad \exp(-L(\mathbf{s}_{MP}))(2\pi)^{P/2}\det(\mathbf{H})^{-1/2} \\ &= \left(\frac{2\pi}{\beta}\right)^{-N/2}\prod_{i=1}^{P}\alpha_i^{1/2}\exp(-L(\mathbf{s}_{MP}))\det(\mathbf{H})^{-1/2}. \end{aligned} \tag{4.52}$$

The natural logarithm of the evidence is then

$$\begin{aligned}\ln p(\mathbf{m}|\mathbf{\Lambda},\beta) &= -\frac{N}{2}\ln\frac{2\pi}{\beta}+\frac{1}{2}\ln\prod_{i=1}^{P}\alpha_i - L(\mathbf{s}_{MP})-\frac{1}{2}\ln\det(\mathbf{H})\\ &= -\frac{N}{2}\ln\frac{2\pi}{\beta}+\frac{1}{2}\sum_{i=1}^{P}\ln\alpha_i - L(\mathbf{s}_{MP})-\frac{1}{2}\ln\det(\mathbf{H}).\end{aligned} \quad (4.53)$$

Lets first find the most probable value of each diagonal element of $\mathbf{\Lambda}$ by differentiating with respect to α_k $(k = 1, ..., P)$.

$$\begin{aligned}\frac{\partial}{\partial\alpha_k}\ln p(\mathbf{m}|\mathbf{\Lambda},\beta) &= \frac{1}{2\alpha_k}-\frac{\partial}{\partial\alpha_k}L(\mathbf{s}_{MP})-\frac{1}{2}\frac{\partial}{\partial\alpha_k}\ln\det(\mathbf{H})\\ &= \frac{1}{2\alpha_k}-\frac{\partial}{\partial\alpha_k}\left(\frac{\beta}{2}|\mathbf{m}-\mathbf{A}\mathbf{s}_{MP}|^2+\frac{1}{2}\sum_{i=1}^{P}\alpha_i s_i^2\right)\\ &\quad -\frac{1}{2}\frac{\partial}{\partial\alpha_k}\ln\det(\mathbf{H})\\ &= \frac{1}{2\alpha_k}-\frac{s_k^2}{2}-\frac{1}{2}\frac{\partial}{\partial\alpha_k}\ln\det(\mathbf{H}).\end{aligned} \quad (4.54)$$

The last derivative on the right hand side of the equation above is a little bit more tricky. In appendix A.4 it is calculated and using the results derived there gives

$$\frac{\partial}{\partial\alpha_k}\ln p(\mathbf{m}|\mathbf{\Lambda},\beta) = \frac{1}{2\alpha_k}-\frac{s_k^2}{2}-\frac{h'_{kk}}{2} \quad (4.55)$$

where h'_{kk} is the k-th diagonal element of the inverse Hessian matrix:

$$\mathbf{H}^{-1} = \left(\beta\mathbf{A}^T\mathbf{A}+\mathbf{\Lambda}\right)^{-1}.$$

Setting the derivative with respect to α_k to zero gives the following identity for α_k which maximizes the evidence

$$\alpha_k = \frac{1}{s_k^2+h'_{kk}}. \quad (4.56)$$

Now an identity for β is needed. Differentiation the log-evidence with respect to β gives

$$\begin{aligned}\frac{\partial}{\partial\beta}\ln p(\mathbf{m}|\mathbf{\Lambda},\beta) &= \frac{N}{2\beta}-\frac{\partial}{\partial\beta}L(\mathbf{s}_{MP})-\frac{1}{2}\frac{\partial}{\partial\beta}\ln\det(\mathbf{H})\\ &= \frac{N}{2\beta}-\frac{1}{2}|\mathbf{m}-\mathbf{A}\mathbf{s}_{MP}|^2-\frac{1}{2}\frac{\partial}{\partial\beta}\ln\det(\mathbf{H})\\ &= \frac{N}{2\beta}-\frac{1}{2}|\mathbf{m}-\mathbf{A}\mathbf{s}_{MP}|^2-\frac{1}{2}\mathrm{Tr}\left(\mathbf{H}^{-1}\frac{\partial}{\partial\beta}\mathbf{H}\right)\\ &= \frac{N}{2\beta}-E_m(\mathbf{s}_{MP})-\frac{1}{2}\mathrm{Tr}\left(\mathbf{H}^{-1}\mathbf{A}^T\mathbf{A}\right).\end{aligned} \quad (4.57)$$

Setting the derivative to zero to find the maximum and writing up the Hessian gives

$$\frac{N}{2\beta} - E_m(\mathbf{s}_{MP}) - \frac{1}{2}\mathrm{Tr}\left((\beta\mathbf{A}^T\mathbf{A} + \mathbf{\Lambda})^{-1}\mathbf{A}^T\mathbf{A}\right) = 0$$

which, with some basic algebra, can be written as

$$2\beta E_m(\mathbf{s}_{MP}) = N - \mathrm{Tr}\left((\mathbf{A}^T\mathbf{A} + \frac{1}{\beta}\mathbf{\Lambda})^{-1}\mathbf{A}^T\mathbf{A}\right). \tag{4.58}$$

Next step is using the equations above in an iterative algorithm to estimate $\mathbf{s}_{MP}$, the diagonal elements α_k of $\mathbf{\Lambda}$ and β.

4.2.2 Algorithm II for Parameter Estimation

Here an iterative algorithm will be presented that uses the equations already established to optimize the hyperparameters $\mathbf{\Lambda}$ and β in the ARD model while also estimating the optimum source vector $\mathbf{s}_{MP}$.

Algorithm II

(i) Initialize $\mathbf{\Lambda}$ and β with positive random numbers. β is drawn from a uniform distribution between 0 and 0.1 while the elements of $\mathbf{\Lambda}$ are drawn from a uniform distribution between 0 and 1.

(ii) Estimate $\mathbf{s}_{MP}$ using equation 4.49:

$$\mathbf{s}_{MP} = \left(\mathbf{A}^T\mathbf{A} + \frac{1}{\beta}\mathbf{\Lambda}\right)^{-1}\mathbf{A}^T\mathbf{m}.$$

(iii) Re-estimate diagonal elements α_k $(k = 1, ..., P)$ of $\mathbf{\Lambda}$ and β using the criteria from equations 4.56 and 4.58:

$$\begin{aligned} \alpha_{k,new} &= \frac{1}{s_k^2 + h'_{kk}} \\ \beta_{new} &= \frac{N - \mathrm{Tr}\left((\mathbf{A}^T\mathbf{A} + \frac{1}{\beta}\mathbf{\Lambda})^{-1}\mathbf{A}^T\mathbf{A}\right)}{2E_m(\mathbf{s}_{MP})} \end{aligned}$$

where $\mathbf{s}_{MP} = [s_1, ..., s_k, ..., s_P]^T$ and h'_{kk} is the k-th diagonal element of the inverse Hessian matrix.

(iv) If convergence criterion fulfilled then stop, else go back to step (ii).

It can be shown that the update equation for α_k in step (iii) is multiplicative (see appendix A.5). The convergence criterion used here is on the maximum change in α_k values that have not reached infinity, and by infinity we mean a predetermined maximum α_k value. Note that α_k should ideally converge to infinity for all non-relevant classes, thus for numerical reasons it is convenient to set a maximum value α_{k-max}. Iteration then stops at step i if $K(i) < \varepsilon$ where

$$K(i) = \max\left\{|\alpha_1(i) - \alpha_1(i-1)|, |\alpha_2(i) - \alpha_2(i-1)|, ..., |\alpha_P(i) - \alpha_P(i-1)|\right\}. \tag{4.59}$$

and $\varepsilon > 0$. It turns out that the algorithm is sensitive to the dimensions of N and P in the model. Meaning that as the model becomes more ill-posed ($N << P$) the value of β becomes worse in the sense that the noise can be under- or over-estimated. Underestimation of noise leads to worse generalization because of overfitting to the noise. Alternatively overestimating the noise inevitably gives worse source estimate. This is the main reason for omitting the change in β from the convergence criterion above and defining a maximum value for β. The maximum value is set by considering a minimum possible SNR ratio. Lets write the SNR ratio given in equation 1.3 for our model $\mathbf{m} = \mathbf{m}_0 + \epsilon$,

$$SNR = \frac{\sigma^2_{m_0}}{\sigma^2_\epsilon}$$

In next section we will see that β should converge toward the noise variance, which is described by the likelihood, i.e. $\sigma^2_\epsilon = \sigma^2_{p(m|s)} = \frac{1}{\beta}$. Using this we can write

$$SNR = \beta\sigma^2_{m_0}.$$

In the worst case $SNR \longrightarrow 0^+$ and the noise is much larger than the signal ($\sigma^2_\epsilon >> \sigma^2_{m_0}$). In this case the β is small and converges usually nicely but estimating the prior correctly is much harder. As the noise gets smaller (i.e. β gets larger) there is a tendency in the algorithm to overfit to the data giving too large value for β. An upper limit is therefore set on SNR which will be called SNR_{max}. This upper limit therefore creates a boundary for β giving

$$\beta_{max} = \frac{SNR_{max}}{\sigma^2_{m_0}}.$$

If we assume $SNR_{max} >> 0$ then $\sigma^2_{m_0} >> \sigma_\epsilon$ and $\sigma^2_{m_0} \simeq \sigma^2_m$ is a good approximation giving

$$\beta_{max} = \frac{SNR_{max}}{\sigma^2_m}. \tag{4.60}$$

Expressing β_{max} in terms of σ_m is preferred since σ_m can easily be estimated from the data. A typically $SNR_{max} = 1000 = 30dB$ is used in chapters 5 and 6. The issue of bad β estimate is also the main reason for introducing another algorithm later on which will be a combination of Algorithm I and II.

4.2.3 Performance Evaluation

Like in Algorithm I the β value represents the inverse of the noise variance and should converge toward it,

$$\sigma^2_{p(m|s)} = \frac{1}{\beta}. \tag{4.61}$$

The α_k values represent similarly the variances of the classes. Remember that each source s_k ($k = 1, ..., P$) belongs to its own class. If we let σ_k^2 denote the variance of each class k the converged α_k values should fulfill

$$\sigma_k^2 = \frac{1}{\alpha_k}. \tag{4.62}$$

This means that if source s_k contains no signal the variance should be zero giving $\alpha_k \longrightarrow \infty$ as the algorithm converges. This can be interpreted in the following way, as the signal (magnitude) of source s_k decreases toward zero the relevance parameter α_k belonging to that source should increase and approach infinity. So if the source vector $\mathbf{s}$ is sparse then many α_k values should approach infinity. The convergence of Algorithm II can therefore not be evaluated in exactly the same way as was done in Algorithm I.

But as in Algorithm I the non-negativity of the hyperparameters update equations is of importance. Lets first take a look at the update equation for α_k ($k = 1, ..., P$), namely

$$\alpha_{k,new} = \frac{1}{s_k^2 + h'_{kk}}.$$

Obviously $s_k^2 \geq 0$ for all $k = 1, ...P$, the necessary condition for a positive $\alpha_{k,new}$ is therefore $h'_{kk} > 0$, where h'_{kk} is the k-th diagonal element of the inverse Hessian matrix $\mathbf{H}^{-1} = \left(\beta \mathbf{A}^T\mathbf{A} + \mathbf{\Lambda}\right)^{-1}$. The matrix $\beta\mathbf{A}^T\mathbf{A}$ is real square symmetric so the Hessian matrix is also real square symmetric which means $\mathbf{H} = \mathbf{H}^T$. The inverse of a symmetric matrix is also symmetric and the eigenvectors of real symmetric matrices can be chosen to be orthogonal, so if $\mathbf{U}$ is a $P \times P$ matrix with the eigenvectors $\mathbf{u}_i$ of $\mathbf{H}$ as the columns then $\mathbf{U}$ is orthogonal, i.e.

$$\mathbf{U}^T\mathbf{U} = \mathbf{I}.$$

Let λ_i be the eigenvalues of $\mathbf{H}$, then by forming a diagonal matrix $\mathbf{E}$ with the eigenvalues on the diagonal the eigenvector equation $\mathbf{H}\mathbf{u}_i = \lambda_i \mathbf{u}_i$ can be written as

$$\mathbf{H}\mathbf{U} = \mathbf{U}\mathbf{E}.$$

Then using the orthogonality of $\mathbf{U}$ we get

$$\mathbf{H} = \mathbf{U}\mathbf{E}\mathbf{U}^T.$$

Taking the inverse and using the orthogonality again ($\mathbf{U}^T = \mathbf{U}^{-1}$) we get

$$\begin{aligned}\mathbf{H}^{-1} &= (\mathbf{UEU}^T)^{-1} \\ &= (\mathbf{U}^T)^{-1}\mathbf{E}^{-1}\mathbf{U}^{-1} \\ &= \mathbf{UE}^{-1}\mathbf{U}^T\end{aligned}$$

which can equivalently be written

$$\mathbf{H}^{-1} = \sum_{i=1}^{P} \frac{1}{\lambda_i}\mathbf{u}_i\mathbf{u}_i^T. \tag{4.63}$$

If we let $\mathbf{u}_i\hat{\ }2$ denote the elements vise square of $\mathbf{u}_i$ then the diagonal elements of the inverse Hessian can be written

$$\begin{bmatrix} h'_{11} \\ \vdots \\ h'_{PP} \end{bmatrix} = \sum_{i=1}^{P} \frac{1}{\lambda_i}\mathbf{u}_i\hat{\ }2.$$

Which shows that the sign of h'_{kk} only depends on the signs of the eigenvalues of the Hessian. The final step is therefore to show that the eigenvalues of the Hessian are positive, or equivalently that the Hessian is positive definite. For information about positive definite matrices see e.g. [32]. The matrix $\beta\mathbf{A}^T\mathbf{A}$ is positive semi-definite, i.e. for all $\mathbf{x} \neq \mathbf{0}$ then

$$\mathbf{x}^T(\beta\mathbf{A}^T\mathbf{A})\mathbf{x} \geq 0.$$

Lets then check if the Hessian is positive definite

$$\mathbf{x}^T\left(\beta\mathbf{A}^T\mathbf{A} + \mathbf{\Lambda}\right)\mathbf{x} = \mathbf{x}^T(\beta\mathbf{A}^T\mathbf{A})\mathbf{x} + \mathbf{x}^T\mathbf{\Lambda}\mathbf{x}$$

where

$$\begin{aligned}\mathbf{x}^T\mathbf{\Lambda}\mathbf{x} &= [x_1, x_2, ..., x_P]\mathbf{\Lambda}\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_P \end{bmatrix} \\ &= \alpha_1 x_1^2 + \alpha_2 x_2^2 + ... + \alpha_P x_P^2.\end{aligned}$$

Therefore if $\alpha_k > 0$ for all $k = 1, ...P$ then

$$\mathbf{x}^T\mathbf{\Lambda}\mathbf{x} > 0$$

and the Hessian is positive definite, meaning that its eigenvalues are positive giving positive update of $\alpha_{k,new}$. We can therefore conclude that if we make sure to initialize all $\alpha_k > 0$ the update equation will always give positive $\alpha_{k,new}$, i.e.

$$\alpha_{k,new} > 0. \tag{4.64}$$

Next lets take a look at the update equation for β:

$$\beta_{new} = \frac{N - \mathrm{Tr}\left((\mathbf{A}^T\mathbf{A} + \frac{1}{\beta}\mathbf{\Lambda})^{-1}\mathbf{A}^T\mathbf{A}\right)}{2E_m(\mathbf{s}_{MP})}$$

$E_m(\mathbf{s}_{MP})$ is non-negative by definition. The necessary condition for a non-negative β is therefore to show that

$$0 \leq N - \mathrm{Tr}\left((\mathbf{A}^T\mathbf{A} + \frac{1}{\beta}\mathbf{\Lambda})^{-1}\mathbf{A}^T\mathbf{A}\right)$$

applies. By initializing the hyperparameters with positive numbers the diagonal matrix $\frac{1}{\beta}\mathbf{\Lambda}$ has positive numbers on its diagonal. Taking the square root of its elements can therefore be done and we can write

$$\frac{1}{\beta}\mathbf{\Lambda} = \left(\frac{\mathbf{\Lambda}}{\beta}\right)^{1/2}\left(\frac{\mathbf{\Lambda}}{\beta}\right)^{1/2}.$$

Using this in the trace expression gives

$$\begin{aligned}
&\mathrm{Tr}\left((\mathbf{A}^T\mathbf{A} + \frac{1}{\beta}\mathbf{\Lambda})^{-1}\mathbf{A}^T\mathbf{A}\right)\\
= \;&\mathrm{Tr}\left(\left(\frac{\mathbf{\Lambda}}{\beta}\right)^{-1/2}\left(\left(\frac{\mathbf{\Lambda}}{\beta}\right)^{-1/2}\mathbf{A}^T\mathbf{A}\left(\frac{\mathbf{\Lambda}}{\beta}\right)^{-1/2} + \mathbf{I}\right)^{-1}\left(\frac{\mathbf{\Lambda}}{\beta}\right)^{-1/2}\mathbf{A}^T\mathbf{A}\right)\\
= \;&\mathrm{Tr}\left(\left(\left(\frac{\mathbf{\Lambda}}{\beta}\right)^{-1/2}\mathbf{A}^T\mathbf{A}\left(\frac{\mathbf{\Lambda}}{\beta}\right)^{-1/2} + \mathbf{I}\right)^{-1}\left(\frac{\mathbf{\Lambda}}{\beta}\right)^{-1/2}\mathbf{A}^T\mathbf{A}\left(\frac{\mathbf{\Lambda}}{\beta}\right)^{-1/2}\right)\\
= \;&\mathrm{Tr}\left(\left(\mathbf{Q}^T\mathbf{Q} + \mathbf{I}\right)^{-1}\mathbf{Q}^T\mathbf{Q}\right)
\end{aligned}$$

where $\mathbf{Q} = \mathbf{A}(\mathbf{\Lambda}/\beta)^{-1/2}$ has been introduced to simplify the notation. The matrix $\mathbf{Q}$ is of same dimensions as $\mathbf{A}$ $(N \times P)$ and remember that $\mathrm{eig}(\mathbf{A}^T\mathbf{A}) = \mathrm{eig}(\mathbf{A}\mathbf{A}^T)$. $\mathbf{Q}$ therefore has N eigenvalues $\lambda_n \geq 0$ and the trace can be written

$$\mathrm{Tr}\left((\mathbf{A}^T\mathbf{A} + \frac{1}{\beta}\mathbf{\Lambda})^{-1}\mathbf{A}^T\mathbf{A}\right) = \sum_{n=1}^{N}\frac{\lambda_n}{\lambda_n + 1} \leq N$$

or equivalently

$$N - \mathrm{Tr}\left((\mathbf{A}^T\mathbf{A} + \frac{1}{\beta}\mathbf{\Lambda})^{-1}\mathbf{A}^T\mathbf{A}\right) \geq 0.$$

We can therefore conclude that

$$\beta_{k,new} \geq 0. \tag{4.65}$$

4.2.4 Numerical Issues

For practical applications where $N << P$ one faces some problems when implementing the algorithm on a computer. In step (ii) of the algorithm the inverse of a $P \times P$ matrix has to be calculated, as was also the case in Algorithm I but there the CPU intensity was minimized using SVD. Here some improvements can be obtained using some matrix algebra. In [31] the following identity is listed

$$(\mathbf{C} + \mathbf{B}\mathbf{B}^T)^{-1}\mathbf{B}^T = \mathbf{C}^{-1}\mathbf{B}(\mathbf{I} + \mathbf{B}^T\mathbf{C}^{-1}\mathbf{B})^{-1} \tag{4.66}$$

for matrices $\mathbf{B}$ and $\mathbf{C}$. This can be used on the identity for calculating $\mathbf{s}_{MP}$

$$\begin{aligned}
\mathbf{H}^{-1}\mathbf{A}^T &= \left(\beta\mathbf{A}^T\mathbf{A} + \mathbf{\Lambda}\right)^{-1}\mathbf{A}^T \\
&= \frac{1}{\beta}\left(\frac{1}{\beta}\mathbf{\Lambda} + \mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T \\
&= \mathbf{\Lambda}^{-1}\mathbf{A}^T\left(\mathbf{I} + \beta\mathbf{A}\mathbf{\Lambda}^{-1}\mathbf{A}^T\right)^{-1} \\
&= \frac{1}{\beta}\mathbf{\Lambda}^{-1}\mathbf{A}^T\left(\frac{1}{\beta}\mathbf{I} + \mathbf{A}\mathbf{\Lambda}^{-1}\mathbf{A}^T\right)^{-1}
\end{aligned}$$

Looking at the final identity on the right the inverse needed is that of an $N \times N$ matrix instead of the original, usually much bigger, $P \times P$ Hessian matrix (remember that the inverse of the diagonal matrix $\mathbf{\Lambda}$ can be obtained by taking the scalar inverse of the diagonal elements). Lets introduce the notation

$$\mathbf{T} = \mathbf{H}^{-1}\mathbf{A}^T = \frac{1}{\beta}\mathbf{\Lambda}^{-1}\mathbf{A}^T\left(\frac{1}{\beta}\mathbf{I} + \mathbf{A}\mathbf{\Lambda}^{-1}\mathbf{A}^T\right)^{-1}. \tag{4.67}$$

Then the most probable source vector can be written

$$\mathbf{s}_{MP} = \beta\mathbf{T}\mathbf{m}. \tag{4.68}$$

Re-estimation of hyperparameter β in step (iii) can furthermore be written

$$\begin{aligned}
\beta_{new} &= \frac{N - \mathrm{Tr}\left((\mathbf{A}^T\mathbf{A} + \frac{1}{\beta}\mathbf{\Lambda})^{-1}\mathbf{A}^T\mathbf{A}\right)}{2E_m(\mathbf{s}_{MP})} \\
&= \frac{N - \mathrm{Tr}(\beta\mathbf{T}\mathbf{A})}{2E_m(\mathbf{s}_{MP})}.
\end{aligned} \tag{4.69}$$

Note that the trace of a matrix is simply the sum of the diagonal elements meaning that instead of calculating the full matrix $\beta\mathbf{TA}$ we only need to find the diagonal elements. Now the re-estimation of α_k parameters remains and that involves using the diagonal elements of the inverse Hessian matrix h'_{kk} directly.

We can use the so called *Kailath variant* [31], which states that for matrices $\mathbf{E}$, $\mathbf{F}$ and $\mathbf{G}$ the following holds

$$(\mathbf{E}+\mathbf{F}\mathbf{G})^{-1} = \mathbf{E}^{-1} - \mathbf{E}^{-1}\mathbf{F}(\mathbf{I}+\mathbf{G}\mathbf{E}^{-1}\mathbf{F})^{-1}\mathbf{G}\mathbf{E}^{-1}.$$

Lets use this identity on the inverse Hessian matrix

$$\begin{aligned}
\mathbf{H}^{-1} &= (\beta\mathbf{A}^T\mathbf{A}+\boldsymbol{\Lambda})^{-1} \\
&= \frac{1}{\beta}(\frac{1}{\beta}\boldsymbol{\Lambda}+\mathbf{A}^T\mathbf{A})^{-1} \\
&= \frac{1}{\beta}\left(\beta\boldsymbol{\Lambda}^{-1} - \beta\boldsymbol{\Lambda}^{-1}\mathbf{A}^T(\mathbf{I}+\beta\mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^T)^{-1}\mathbf{A}\beta\boldsymbol{\Lambda}^{-1}\right) \\
&= \boldsymbol{\Lambda}^{-1} - \boldsymbol{\Lambda}^{-1}\mathbf{A}^T(\frac{1}{\beta}\mathbf{I}+\mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^T)^{-1}\mathbf{A}\boldsymbol{\Lambda}^{-1} \qquad (4.70)
\end{aligned}$$

so instead of having to take the inverse of the $P \times P$ Hessian matrix we only have to calculate the inverse of an $N \times N$ matrix and a diagonal matrix. Looking at the final result here above we see that in the matrix multiplication furthest to the right the first and last matrices are diagonal which means that we do not have to do the full matrix multiplication. We can simply calculate the diagonal of the matrix $\mathbf{A}^T(\frac{1}{\beta}\mathbf{I}+\mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^T)^{-1}\mathbf{A}$. Lets call its k-th diagonal element t_{kk} and use it with the k-th diagonal element of $\boldsymbol{\Lambda}$ to calculate the k-th diagonal element h'_{kk} of the inverse Hessian, which is needed for the re-estimation $\alpha_{k,new}$, i.e. we calculate

$$h'_{kk} = \frac{1}{\alpha_k} - \frac{t_{kk}}{\alpha_k^2} = \frac{1}{\alpha_k}\left(1-\frac{t_{kk}}{\alpha_k}\right). \qquad (4.71)$$

Calculating only the diagonal elements t_{kk} can be implemented very efficiently on a computer and a proposed Matlab solution can be seen in appendix B.

4.2.5 Algorithm IIb

As was briefly noted before the convergence of β in Algorithm II is is sensitive to how ill-posed the system is, meaning that as P becomes larger than N the β value returned by the algorithm gets worse with respect to the generalization. This was not the case with Algorithm I and the β value returned was not as sensitive to the dimensions N and P. Here an algorithm is therefore presented that takes advantage of that. It first uses Algorithm I to find the β value and then uses Algorithm II to find the diagonal values of Λ and the source estimates $\mathbf{s}_{MP}$.

Algorithm IIb

(i) Initialize $\mathbf{\Lambda}$ with positive random numbers between 0 and 1 drawn from a uniform distribution.

(ii) Run Algorithm I and use the β value returned. β is now constant and the equations from Algorithm II are used to find $\mathbf{\Lambda}$ and $\mathbf{s}_{MP}$.

(iii) Estimate $\mathbf{s}_{MP}$ using equation 4.49:

$$\mathbf{s}_{MP} = \left(\mathbf{A}^T\mathbf{A} + \frac{1}{\beta}\mathbf{\Lambda}\right)^{-1} \mathbf{A}^T\mathbf{m}.$$

(iv) Re-estimate diagonal elements α_k $(k = 1, ..., P)$ of $\mathbf{\Lambda}$ using the criterion from equation 4.56:

$$\alpha_{k,new} = \frac{1}{s_k^2 + h'_{kk}}$$

where $\mathbf{s}_{MP} = [s_1, ...s_k, ...s_P]^T$ and h'_{kk} is the k-th diagonal element of the inverse Hessian matrix.

(v) If convergence criterion fulfilled then stop, else go back to step (iii).

The convergence criterion used in step (v) is the same as is used in Algorithm II. In the ideal case both Algorithm II and IIb should give the same results since the only difference between them is how the noise is estimated. In practice however the noise estimate in Algorithm IIb is better resulting in better source estimates.

4.2.6 Active Sets (Algorithm IIc)

When using automatic relevance determination (ARD) it was noted before that for irrelevant sources the corresponding hyperparameter α_k converges to infinity, effectively removing the source from the model. This can be used to gradually decrease the size of the model during the iteration procedure, significantly reducing the computation for large models. By applying this to Algorithm IIb the same update equations can be used but an extra step has to be added to the algorithm where classes that have exceeded the α_{k-max} value are removed from the model. This step would come between steps (iv) and (v) in Algorithm IIb, i.e. after updating the α_k parameters we would remove the ones from the model that have exceeded α_{k-max}. Lets attempt to explain this a little bit more in detail. Assume the model is unchanged before iteration i. After updating the parameters then j number of α_k values have exceeded α_{k-max}. Those j classes are removed from the model, i.e. j sources and their hyperparameters are removed. This effectively shrinks the matrix $\mathbf{A}$ down to $N \times (P-j)$, the source vector $\mathbf{s}$ down to $(P-j)$ and the diagonal matrix $\mathbf{\Lambda}$ down to $(P-j) \times (P-j)$. As more classes are removed during the iteration process the system shrinks more and more ideally leaving only the relevant sources. This is fairly easy to add to Algorithm IIb and only requires some additional book keeping during the iteration process to keep track of which indices of the original classes are still relevant at the end of the iteration. For simulation purposes **Algorithm IIc** refers to Algorithm IIb with the active set feature added, so that inactive sets are removed from the model during the iteration.

4.2.7 Low Pass Filtering (Algorithm IId)

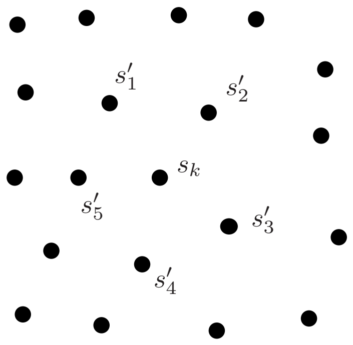

Figure 4.2: *The dots represent a grid of sources. Source s_k is labeled and its nearest neighbours s'_j (j=1,...,5).*

In general one would expect the activity over the cortical surface to be smooth in the sense that if a source is active then its neighboring sources are more likely also to be active than other sources placed further away. This relates directly to the fact the EEG signals are believed to be generated by coherent groups of pyramidal neurons. There is an intuitive way to incorporate spatial smoothing on the source magnitudes, we simply add a low pass filter step into the iteration procedure of Algorithm IIb between steps (iii) and (iv) by applying a window function on the sources. Lets formulate this in a mathematical way. Assume each source s_k has n_k number of nearest neighbours. The neighbours of s_k will be denoted s'_j $(j = 1, ..., n_k)$. Then for $q \in]0;1[$ the filtered source s_{LPk} is

$$s_{LPk} = q s_k + \frac{1-q}{n_k} \sum_{j=1}^{n_k} s'_j. \tag{4.72}$$

Figure 4.2 shows a grid of discrete sources, these could represent sources placed on the nodes of the cortical tesselation surface. A source s_k is labeled on the figure along with its neighbouring sources. Note that the distance between the sources is not taken into account in the filter. This low passed version of Algorithm IIb shall be called **Algorithm IId** from now on. In the following section an attempt is made to incorporate spatial smoothing into the prior distribution.

4.3 Smoothing Prior

In section 4.2 an ARD framework was presented with a prior distribution suitable for sparse source space. The sparsity assumption is reasonable for some cases of EEG measurements (e.g. averaged EPs), but furthermore one would expect the activity in these cases to be localized in the sense that neighbouring neurons are more likely to fire than ones placed further apart. With this in mind spatial smoothing in the form of an average window function was added to the ARD algorithm in section 4.2.7. Here an attempt will be made to incorporate spatial smoothing into the prior distribution itself. So instead of applying a single hyperparameter to each source a neighborhood of sources will be given a hyperparameter α_k. The neighborhood is defined in a similar way as was done in section 4.2.7 and shown on figure 4.2. For each source and its nearest neighbours a class is defined where the center source has the highest weight. The source space (i.e. the cortex) will in this way be divided into overlapping classes centered around each source. General notation for an ARD prior was given with equation 4.42 and in terms of the overlapping classes we define the prior as

$$p(\mathbf{s}) = \frac{1}{Z_s(\alpha_1, ..., \alpha_P)} \exp\left(-\frac{1}{2}\sum_{i=1}^{P}\alpha_i\left(s_i^2 + \frac{1}{n_i}\sum_{j=1}^{n_i}(s'_j)^2\right)\right). \tag{4.73}$$

where for each source s_k $(k = 1, ..., P)$, n_i are the number of neighbouring sources s'_j as explained on figure 4.2. Notice the relation with equation 4.72. The classes will overlap and the total number of classes equal the number of sources. In matrix notation the prior can be written

$$p(\mathbf{s}) = \frac{1}{Z_s(\alpha_1, ..., \alpha_P)} \exp\left(-\frac{1}{2}\sum_{i=1}^{P}\alpha_i \mathbf{s}^T \mathbf{C}_i \mathbf{s}\right). \tag{4.74}$$

where $\mathbf{C}_i$ is a diagonal matrix defining the connectivity for each source s_i. For the following calculations the notation from equation 4.73 will be used which avoids defining different $\mathbf{C}_i$ matrices for each source. One of the key steps in previous sections when going through the Bayesian formulation was finding the normalization factors analytically. This is also possible here and for the prior the normalization factor is

$$
\begin{aligned}
Z_s(\alpha_1, ..., \alpha_P) &= \int_{-\infty}^{\infty} \exp\left(-\frac{1}{2}\sum_{i=1}^{P}\alpha_i\left(s_i^2 + \frac{1}{n_i}\sum_{j=1}^{n_i}(s'_j)^2\right)\right) d\mathbf{s} \\
&= \int_{-\infty}^{\infty} \exp\left(-\frac{1}{2}\sum_{i=1}^{P}(\alpha_i + \psi_i)s_i^2\right) d\mathbf{s}. \qquad (4.75)
\end{aligned}
$$

In this first step the prior has only been rewritten to a simpler form where a new variable ψ_i $(i = 1, ..., P)$ has been defined. Using this new variable ψ_i will make following calculations simpler and more readable so it is important to give a good accurate definition of it here. By looking at the prior in equation 4.73 we see that each hyperparameter α_i is assigned to a source s_i along with an average of its closest neighbouring sources s'_j. This causes overlapping of the classes in the sense that each source does not only have a single hyperparameter as in section 4.2. So instead of writing the prior in terms of how single hyperparameters effect groups of sources the step taken in the equation above rewrites the prior in terms of how single sources s_i are effected by groups of hyperparameters $(\alpha_i + \psi_i)$, so ψ_i defines a group of hyperparameters. Now let α'_j define the hyperparameters centered around the neighbouring sources s'_j of s_i and n'_j be the number of neighbours for each s'_j. Then if source s_i has n_i number of neighbours we get

$$
\psi_i(\alpha'_1, ..., \alpha'_{n_i}, n'_1, ..., n'_{n_i}) = \frac{\alpha'_1}{n'_1} + \frac{\alpha'_2}{n'_2} + ... + \frac{\alpha'_{n_i}}{n'_{n_i}}. \qquad (4.76)
$$

Continuing with the calculation of $Z_s(\alpha_1, ..., \alpha_P)$ is now straight forward and done in the same way as in previous sections,

$$
\begin{aligned}
Z_s(\alpha_1, ..., \alpha_P) &= \prod_{i=1}^{P}\int_{-\infty}^{\infty} \exp\left(-\frac{1}{2}(\alpha_i + \psi_i)s_i^2\right) ds_i \\
&= \prod_{i=1}^{P}\left(\frac{2\pi}{\alpha_i + \psi_i}\right)^{1/2} \\
&= (2\pi)^{P/2}\prod_{i=1}^{P}(\alpha_i + \psi_i)^{-1/2}. \qquad (4.77)
\end{aligned}
$$

The likelihood is the same as before (see equations 4.10 and 4.14) and rewritten here for clarity

$$
p(\mathbf{m}|\mathbf{s}) = \frac{1}{Z_m(\beta)} \exp\left(-\frac{\beta}{2}|\mathbf{m} - \mathbf{A}\mathbf{s}|^2\right)
$$

where

$$
Z_m(\beta) = \left(\frac{2\pi}{\beta}\right)^{N/2}.
$$

Now Bayes' theorem (see eq. 4.3 and 4.4) can be used to form an identity for the posterior distribution:

$$p(\mathbf{s}|\mathbf{m}) = \frac{1}{Z_L(\alpha_1, ..., \alpha_P, \beta)} \exp(-L(\mathbf{s}, \alpha_1, ..., \alpha_P, \beta)) \tag{4.78}$$

with

$$Z_L(\alpha_1, ..., \alpha_P, \beta) = \int \exp(-L(\mathbf{s}, \alpha_1, ..., \alpha_P, \beta)) d\mathbf{s} \tag{4.79}$$

where

$$\begin{aligned} L(\mathbf{s}, \alpha_1, ..., \alpha_P, \beta) &= \frac{\beta}{2}|\mathbf{m} - \mathbf{A}\mathbf{s}|^2 + \frac{1}{2}\sum_{i=1}^{P}(\alpha_i + \psi_i)s_i^2 & (4.80) \\ &= \frac{\beta}{2}|\mathbf{m} - \mathbf{A}\mathbf{s}|^2 + \frac{1}{2}\mathbf{s}^T\mathbf{\Gamma}\mathbf{s}, & (4.81) \end{aligned}$$

and the diagonal matrix $\mathbf{\Gamma}$ is defined

$$\mathbf{\Gamma} = \begin{bmatrix} \alpha_1 + \psi_i & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \alpha_P + \psi_P \end{bmatrix}, \tag{4.82}$$

note the resemblance to the $\mathbf{\Lambda}$ matrix from last section which had only the α's on the diagonal. As before maximizing the posterior to find the most probable value of $\mathbf{s}$ is equivalent to maximizing $-L(\mathbf{s}, \alpha_1, ..., \alpha_P, \beta)$ with respect to $\mathbf{s}$, assuming the hyperparameters are known. With the similarity of $\mathbf{\Lambda}$ and $\mathbf{\Gamma}$ in mind the results from appendix A.3.2 can be used when finding the derivative of L with respect to $\mathbf{s}$ giving

$$\frac{\partial}{\partial \mathbf{s}}(-L(\mathbf{s}, \alpha_1, ..., \alpha_P, \beta)) = \beta(\mathbf{A}^T\mathbf{m} - \mathbf{A}^T\mathbf{A}\mathbf{s}) - \mathbf{\Gamma}\mathbf{s}. \tag{4.83}$$

Setting this equal to zero results in the most probable value

$$\mathbf{s}_{MP} = \left(\mathbf{A}^T\mathbf{A} + \frac{1}{\beta}\mathbf{\Gamma}\right)^{-1}\mathbf{A}^T\mathbf{m}. \tag{4.84}$$

The similarity to equations 4.19 and 4.49 is obvious and enables some intuitive interpretation. We can say that equation 4.49 gives a sparse representation of 4.19 and equation 4.84 is a smoothed version of the sparse case 4.49.

Rewriting $L(\mathbf{s})$ around the most probable value $\mathbf{s}_{MP}$ can be used, as has been done before in previous sections. Using this a framework for the optimum hyperparameters can be formulated and this is what is done in appendix A.6. The equations derived in the appendix are similar to the ones from the ARD formulation (section 4.2.1) but include some tedious summations which are very impractical for computer implementation and more importantly formulating update equations for α_k $(k = 1, ..., P)$ where non-negativity is guaranteed is very hard, if not impossible. What we do here therefore is use equation 4.84 along with the update equations for the hyperparameters from Algorithm IIb to form a spatially smoothed version of Algorithm IIb.

4.3.1 Algorithm III for Parameter Estimation

Here an iterative algorithm will be presented where the framework from last section will be used to estimate the most probable source vector $\mathbf{s}_{MP}$, introducing a low pass filtering effect on the sources. The update equations from Algorithm IIb will however be used for the hyperparameters α_k $(k = 1, ..., P)$ and β (remember that Algorithm IIb uses Algorithm I to find β). This algorithm, which shall be called Algorithm III, is therefore a mixture of two different prior assumption and by comparing equation 4.55 with that of A.17, which should strictly speaking be used to find the optimum α_k values here, we note that some overestimation of α_k values are expected in Algorithm III leading to a bit too strong low pass filtering.

Algorithm III

(i) Initialize all α_k $(k = 1, ..., P)$ as 1.

(ii) Run algorithm I to find β.

(iii) Estimate $\mathbf{s}_{MP}$ using equation 4.84:

$$\mathbf{s}_{MP} = \left(\mathbf{A}^T\mathbf{A} + \frac{1}{\beta}\mathbf{\Gamma}\right)^{-1} \mathbf{A}^T\mathbf{m}.$$

(iv) Re-estimate α_k $(k = 1, ..., P)$ using the criterion from equation 4.56:

$$\alpha_{k,new} = \frac{1}{s_k^2 + h'_{kk}}$$

where $\mathbf{s}_{MP} = [s_1, ...s_k, ...s_P]^T$ and h'_{kk} is the k-th diagonal element of the inverse Hessian matrix from section 4.2, i.e. $\mathbf{H}^{-1} = \left(\beta\mathbf{A}^T\mathbf{A} + \mathbf{\Lambda}\right)^{-1}$, where $\mathbf{\Lambda}$ is a diagonal matrix with α_k as the diagonal elements.

(v) If convergence criterion fulfilled then stop, else go back to step (iii).

Same convergence criterion as in Algorithm II is used. The numerical methods introduced in section 4.2.4 are also valid and can be used here to avoid computational issues. Since the theoretical framework of this algorithm is a mixture of two prior assumptions the performance evaluation is not as clear. However same should apply as before on the noise estimation from the β parameter, i.e. β should converge toward the inverse of the noise variance. The α_k hyperparameters we would expect to converge in a similar way as in Algorithm II, i.e. α_k centered around an active source should converge toward a finite number proportional to the source amplitude and for inactive sources the corresponding α_k value centered around it should converge to infinity. Furthermore, as mentioned above, we would expect some overestimation of α_k due to the mixing of the two different prior assumptions.

4.4 Summary

In this chapter many methods have been introduced in the form of different algorithms. These algorithms will be tested in the following chapters so for clarity the main parts this chapter will be summarized.

The likelihood distribution for all algorithms was Gaussian meaning that the noise is assumed Gaussian. Algorithm I was based on a Gaussian prior distribution assuming Gaussian distribution of sources. Only two hyperparameters describing the variances of sources and noise are therefore used in Algorithm I.

Algorithm II was based on ARD, assuming sparse sources. The prior distribution in this case had a single hyperparameter for each source along with the noise hyperparameter. Algorithm IIb used Algorithm I to get a better estimate of the noise. By keeping track of hyperparameters converging toward infinity the third ARD algorithm variant was introduced. Algorithm IIc removed inactive set during the iteration. Finally spatial smoothing was introduced into Algorithm IIb by low pass filtering the sources during iteration.

In the last part of the chapter spatial smoothing was incorporated into the prior distribution. Algorithm III used this prior assumption to estimate the sources but the hyperparameter estimation from Algorithm IIb was used.

Chapter 5

Simulations on Artificial Data

Here the algorithms will be tested using simulated data. By simulated data we mean that the source vector **s** is created by hand, in the sense that its values are chosen and therefore known. The lead-field matrix is also known and used to warp the source vector into the measurement space. Then some noise is added to create the artificial measured vector **m**. Inverse calculations can then be performed on **m** and the source estimate $\mathbf{s}_{MP}$ compared to the real source vector **s**. The primary benefit of using simulated data is that many aspects of the algorithms can be controlled. That way the behavior of the algorithms can be carefully evaluated while yielding the important advantage that the answer is known in the validation experiment. But while testing with simulated data is necessary it is not sufficient. Additional testing with real data is presented in next chapter.

In the testing presented here the error will be in terms of the mean squared error, often noted *mse* and in this text often noted *msq error*. If **s** is the true source moments and $\mathbf{s}_{MP}$ is the estimated moments the msq error is

$$\text{msq error} = \frac{1}{P}\sum_{i=1}^{P}(s_i - s_{MPi})^2 \tag{5.1}$$

where s_i and s_{MPi} are the elements of **s** and $\mathbf{s}_{MP}$ respectively and P is the length of the vectors.

In the literature EEG inverse methods are often evaluated in terms of *localization error*, defined as the euclidean distance between the true source and estimated source. In this definition lies the assumption that the number of sources is known and for simulated data this is always the case. For equivalent dipole localization, which was briefly explained in the introduction chapter, the sources are represented by few equivalent dipoles (down to a single dipole). In this method the localization is a crucial factor. In the approach here, distributed source imaging, the localization error can be defined as the distance between the maxima of the true and estimated sources. But in this definition lies also the assumption that the number of active areas is known and the size of the estimated area is not important. When testing on real data the number of sources, or active areas, is not known, although some areas may be expected to be active. Furthermore the size of the active are can also be important. Here we therefore use the msq error as our main quality estimate since it unavoidably takes into account the locations of active sources and the number of active sources. It can however be noted that in the simulated examples to come the localization error in terms of distance between maxima is usually very good.

5.1 Algorithm I

One of the main argument for starting the formulation of Algorithm I presented in section 4.1.2 is that given an ill-posed linear model with additive noise the method of least squares fails. This sets a benchmark for the initial testing, i.e. the algorithm should perform better than the method of least squares. The first simulations are therefore simple examples comparing the performance of the two methods. For the sake of clarity lets rewrite the model

$$\mathbf{m} = \mathbf{A}\mathbf{s} + \boldsymbol{\epsilon}$$

where $\mathbf{m}$ is a vector of N measurements, $\mathbf{A}$ is the $N \times P$ lead-field matrix and $\mathbf{s}$ is the vector of P sources to be estimated. The noise vector $\boldsymbol{\epsilon}$ of length N is Gaussian with zero mean. Remember that in the Bayesian framework of the algorithm the prior and the likelihood are assumed Gaussian.

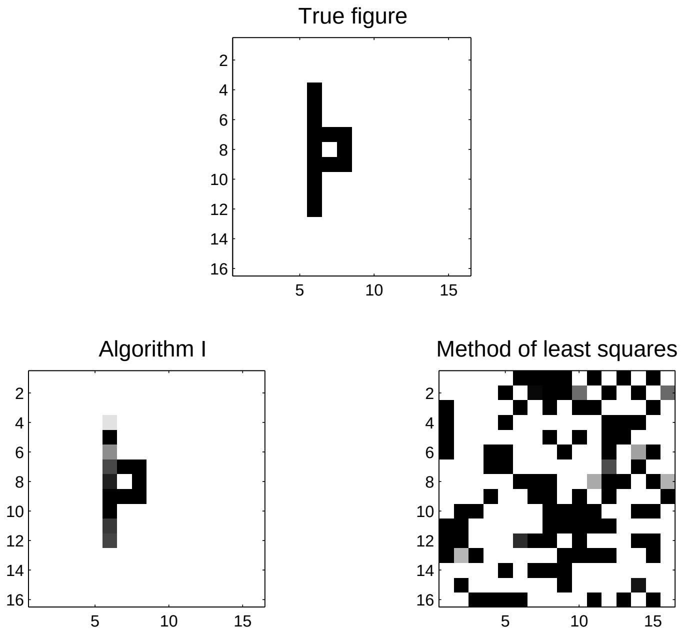

Figure 5.1: *Estimation of source figure using Algorithm I and the method of least squares for a system with $N = 246$ and $P = 256$. The true source is plotted at the top and the bottom two plots show that the algorithm performs much better than the method of least squares.*

5.1.1 Basic Toy Examples

Binary figure

In the first example the source distribution is clearly not Gaussian but it is of interest to look at the example presented on figure 4.1 (page 33) where the method of least squares clearly failed. The 246×256 matrix $\mathbf{A}$ is created by drawing numbers from a Gaussian distribution with zero mean and standard deviation 0.1. Added noise has standard deviation 0.01. Figure 5.1 shows the different results for the algorithm and method of least squares, also plotted is the true figure. The algorithm gives good results here even though the Gaussianity of the prior is clearly broken.

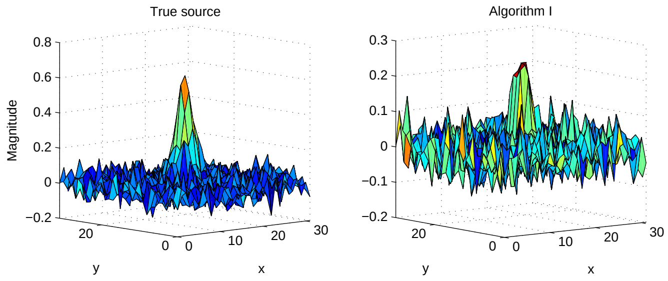

Figure 5.2: *Estimation of source for* **A** *matrix of dimensions* 512 × 1024. *The true source is plotted on the left and the estimate from Algorithm I to the right. In the estimate the pulse is visible.*

Increased activity on Gaussian floor

Here the goal is to estimate a source which is a 2D pulse on Gaussian distributed noise floor. The true source can be seen on the left graph on figure 5.2. This example is more realistic in the sense that it is similar to if one would for example like to detect activity on the motor cortex of subject when moving a limb. That kind of activity one might expect to be a sudden localized increase in activity on the cortex. Here the 32×32 grid could therefore represent the cortical surface and the amplitudes represent the current magnitudes to be estimated. The lead-field matrix $\mathbf{A}$ has dimensions 512×1024 and is generated by drawing numbers from a normal distribution with standard deviation of 0.1. Added measurement noise is Gaussian with zero mean and standard deviation 0.1, giving SNR of 8.0dB. On the graph to the right on figure 5.2 the estimated source is plotted and the pulse is clearly visible, it is also placed at the correct location in the grid.

But how Gaussian is the prior here? Looking at the true source figure one would think that $\mathbf{s}$ is close to Gaussian but not exactly. Figure 5.3 shows a histogram estimation of the distribution of $\mathbf{s}$. Also plotted as a solid curve is a Gaussian distribution with same mean and standard deviation as $\mathbf{s}$. Comparing the two distributions shows that $\mathbf{s}$ is not completely Gaussian. Although not being Gaussian the algorithm converges as can be seen on figure 5.4. Finally lets see how well equations 4.34 and 4.35 hold even though the source distribution here is not Gaussian, we get for the sources (prior)

$$\sigma^2_{p(s)} = 0.0053 \qquad \frac{1}{\alpha} = 0.0061$$

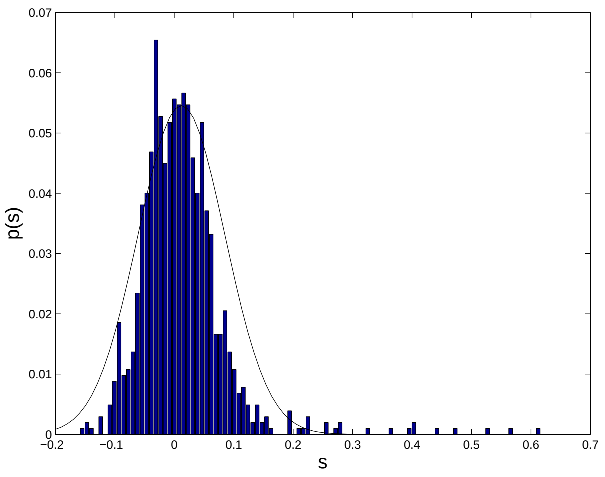

Figure 5.3: *Estimation of the distribution of* $\mathbf{s}$ *(histogram) compared with Gaussian distribution of same mean and standard deviation (solid curve).*

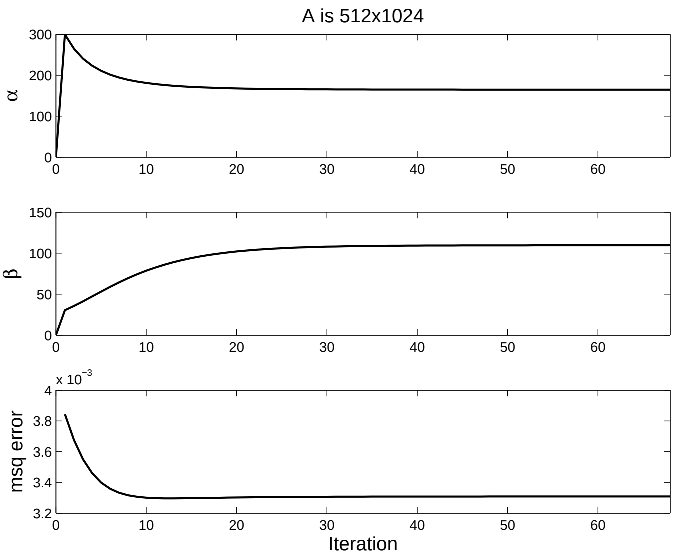

Figure 5.4: *Convergence of* α *and* β *over iterations. Also plotted is the mean squared error of* $\mathbf{s}_{MP}$ *over iterations. The stop threshold for the algorithm was chosen* $\varepsilon = 10^{-3}$.

and for the measurement noise (likelihood)

$$\sigma^2_{p(m|s)} = 0.0100 \qquad \frac{1}{\beta} = 0.0091$$

The values are close to each other and pretty accurate given the normality deviation in the source distribution. Figure 5.4 shows the convergence of α and β over iterations. Also plotted is the mean squared error of the estimated source.

5.1.2 Evaluation of Algorithm I

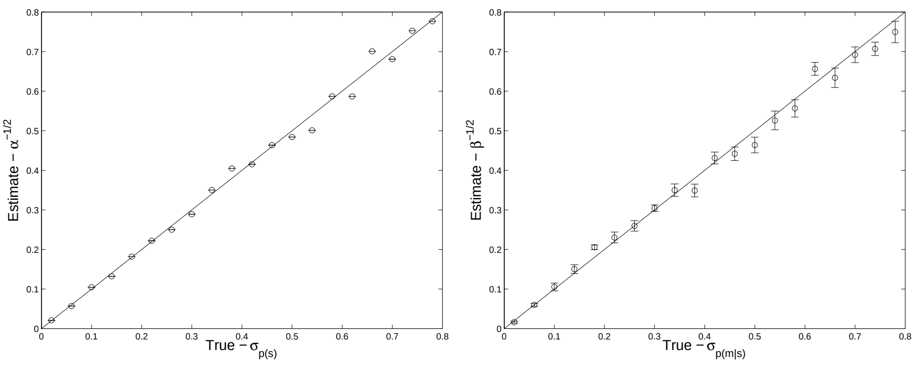

Figure 5.5: *Estimated vs. true variance compared both for prior and likelihood. Each value is calculated using 10 runs with errors bars respectively. When estimating the prior variance the likelihood standard deviation is set to 0.01 and when estimating the likelihood the prior standard deviation is set to 1. That gives similar range of SNR ratio for both tests (10 to 42dB for the α test and 6 to 38dB for the β test). According to equations 4.34 and 4.35 both graphs should be a straight line. The matrix* $\mathbf{A}$ *is created by drawing numbers from a Gaussian distribution with standard deviation 0.1 and the dimensions are* $N = P = 256$.

There is a big unanswered question, given Gaussian prior and likelihood how well does the algorithm perform? The best way to test this is to use equations 4.34 and 4.35. Plotting the estimated noise variance against the true value should give a straight line. On figure 5.5 this is done for both α and β showing that the values lie on a straight line implying good performance of the algorithm. Here the values of the matrix $\mathbf{A}$ are drawn from a Normal distribution with zero mean and 0.1 standard deviation. The error bars for the α values are not visible because of their extremely small values. However the β error bars are visible and they grow smaller as the variance decreases.

There are some limitations when the variance decreases. Figure 5.6 shows how the estimated value converges to a constant value as the variance decreases. Furthermore it should be noted that this test was for a well-posed system of dimensions $N = 256$ and $P = 256$ where the random matrix $\mathbf{A}$ was invertible and non-singular. As the system becomes more ill-posed, i.e. P grows larger than N, the convergence seen on figure 5.6 for the likelihood (graph to the right) starts sooner and the slope on the curve to the right of figure 5.5 starts deviating from unity. This means that the algorithm can under- or overestimate the noise. The deviation is then proportional to the SNR ratio. This effect is shown for a system with $N = 256$ and $P = 1024$ on figure 5.7. There we can see that in

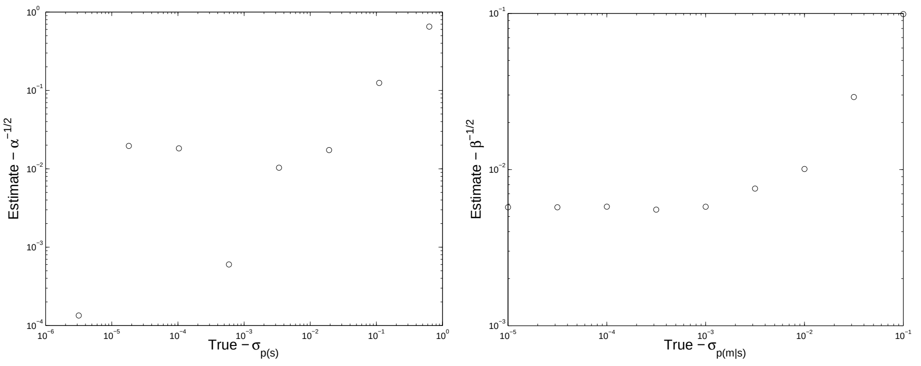

Figure 5.6: *Logarithm of estimated vs. logarithm of true variance for a system of dimensions $N = 256$ and $P = 256$. These curves show for low variances the estimated value converges to a constant.*

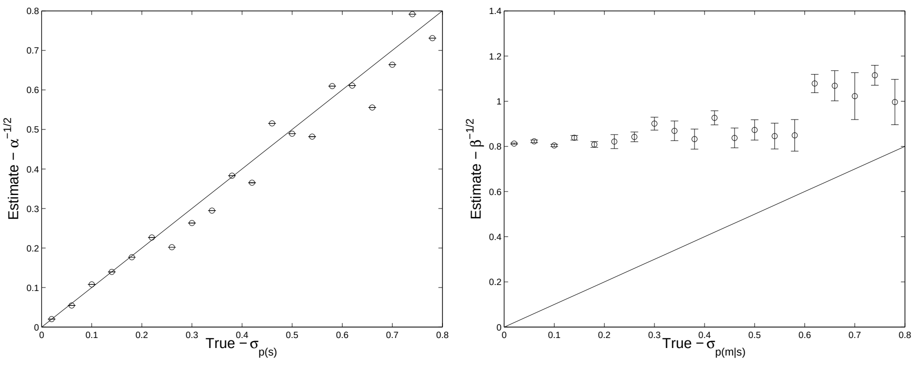

Figure 5.7: *Estimated vs. true variance compared for the prior and likelihood distributions. Each value is calculated using 15 runs with errors bars respectively. The system is ill-posed and the 256×1024 matrix $\mathbf{A}$ is created by drawing numbers from a Gaussian distribution with standard deviation 0.1.*

this case the noise is overestimated and the slope of the true vs. estimate curve is less then unity with some additional bias. The bias can vary for different **A** matrices and although in this case the noise is overestimated that is not the case in general and for some systems the noise is underestimated. But the true vs. estimate curve for the prior is still accurate for this more ill-posed system, there are though a bit more fluctuations in the curve.

Using equation 4.30 the convergence of α and β can be plotted over the surface defined by this equation. Figure 5.8 shows this for a well posed system of dimensions $N = 60$ and $P = 60$. As the graph shows both values converge to approximately 100, which is around the correct standard deviation values of 0.1 for both prior and likelihood. That also amounts to SNR around 0dB with the random matrix **A** used in this example. But as was stated above the estimation of β is more sensitive to the ill-posedness of the system. So what happens to the surface on figure 5.8 as the system becomes more ill-posed? Figure 5.9 shows the same graph but for a more ill-posed system, where $N = 30$ and $P = 70$. On this figure it can be seen how flat the surface is in the β direction. The maximum is therefore not as obvious as on the first graph and β does not converge as close to the true value of 100. This is what happens as the system becomes more ill-posed and also what happens as the SNR ratio becomes worse, although not specifically shown here.

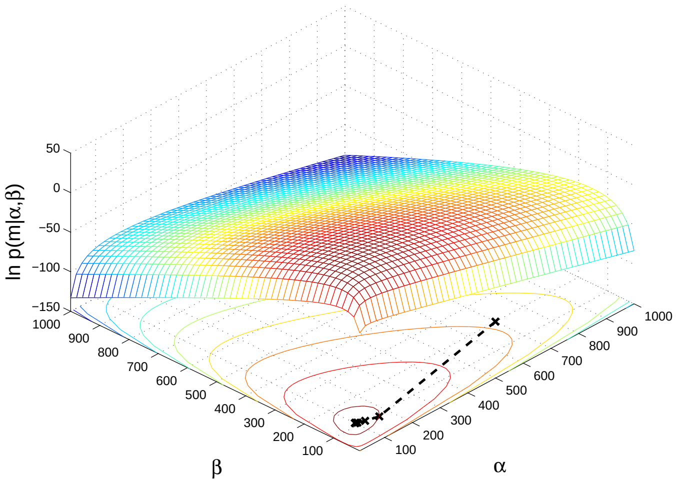

Figure 5.8: *The log-probability* $\ln p(\mathbf{m}|\alpha, \beta)$ *as a function of* α *and* β *for a system with* $N = 60$ *and* $P = 60$. *Also plotted on the contours in the bottom, with black crosses on a dashed curve, is the convergence of* α *and* β. *The first point on the convergence curve is located around* $\alpha = 700$ *and* $\beta = 200$ *and as the iteration progresses the* α *and* β *values converge to 100.*

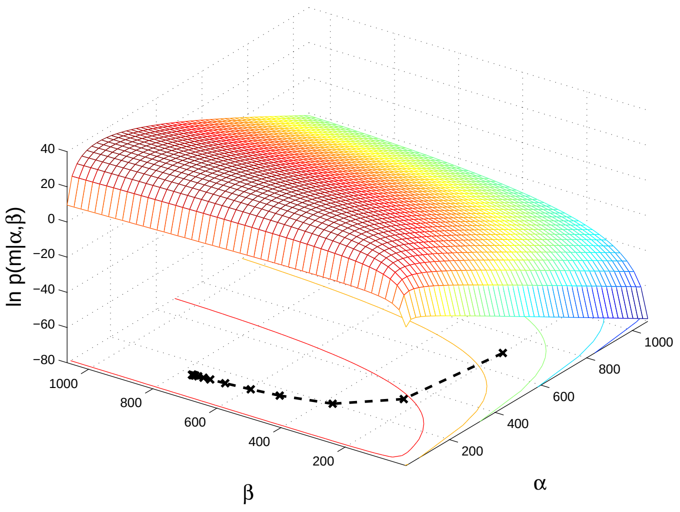

Figure 5.9: *Same as figure 5.8 but for a more ill-posed system, here* $N = 30$ *and* $P = 70$*.* $SNR = 0dB$ *as on figure 5.8. Here* β *converges further from the true value of 100.*

5.1.3 Ill-conditioned 3 Sphere Head Model

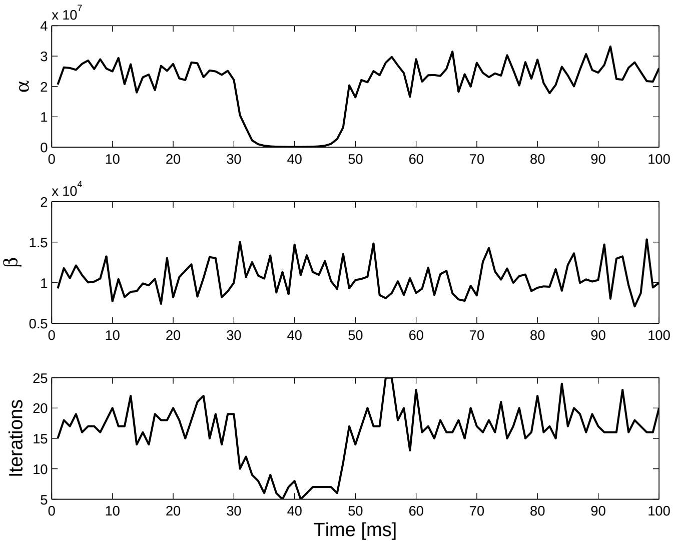

Figure 5.10: *Converged α and β values for each time index. Also plotted is the number of iterations needed at each time index.*

Until now the simulations have showed that the algorithm works and one has to be careful when the Gaussian assumption for the prior is not met. Although the matrix $\mathbf{A}$ has been ill-posed in these examples its dimensions have been lower than is usually the case for real data. Now we take a look at a simulation using a three concentric sphere head model. Spherical models were discussed in section 3.4 and in section 3.5 the software package used for acquiring the forward model, in the form of the lead field matrix $\mathbf{A}$, was introduced. The *3 sphere Berg* BrainStorm routine is used for the modeling. Tesselation surfaces for an imaginary subject are generated using a high density 256 channel EEG electrode grid supplied with the BrainStorm software. And as described in section 3.5 the Montreal Phantom head is warped to match the channel locations. Remember that the sources are localized on the cortical surface with directions perpendicular to the surface. Solving the forward model equations gives the gain matrix $\mathbf{A}$ with $N = 256$ and $P = 10001$. This matrix has condition number $\kappa = 4.54 \times 10^8$ implying a very ill-conditioned matrix. Here the artificial data is some cortical activity over a time period 100ms with time steps of 1ms. This data is similar to the 2D pulse introduced above in the sense that there is some underlying cortical activity which was generated using a Gaussian random number generator, then around time 40ms there is increased activity on the surface. This increased activity is located around the center when the cortex is viewed from the top.

Algorithm I is run for each time instance giving a series of cortical activity.

The final values of α and β are plotted on figure 5.10 for each time instance. Also plotted are the total number of iterations needed to reach the iteration limit of $\varepsilon = 10^{-3}$ (see eq. 4.33). From the graph showing α it can be seen that around time 40 ms when the pulse comes in the estimated α values decrease because the source variance increases. It can be expected that the source becomes less Gaussian distributed at this time. The β value oscillates around 10^{-4} which corresponds to the correct standard deviation value of 0.01 for the added measurement noise. For the underlying Gaussian cortical activity present before and after the pulse the estimated standard deviation from the α value complies with the real value, i.e. both values are around 3.8×10^{-8}. Finally on figure 5.11 the true and estimated sources are plotted for two different time indices. At time 10 ms there is some underlying activity and then at 40 ms the pulse is at maximum amplitude, this is clearly visible from the figures and the estimates resemble the true sources.

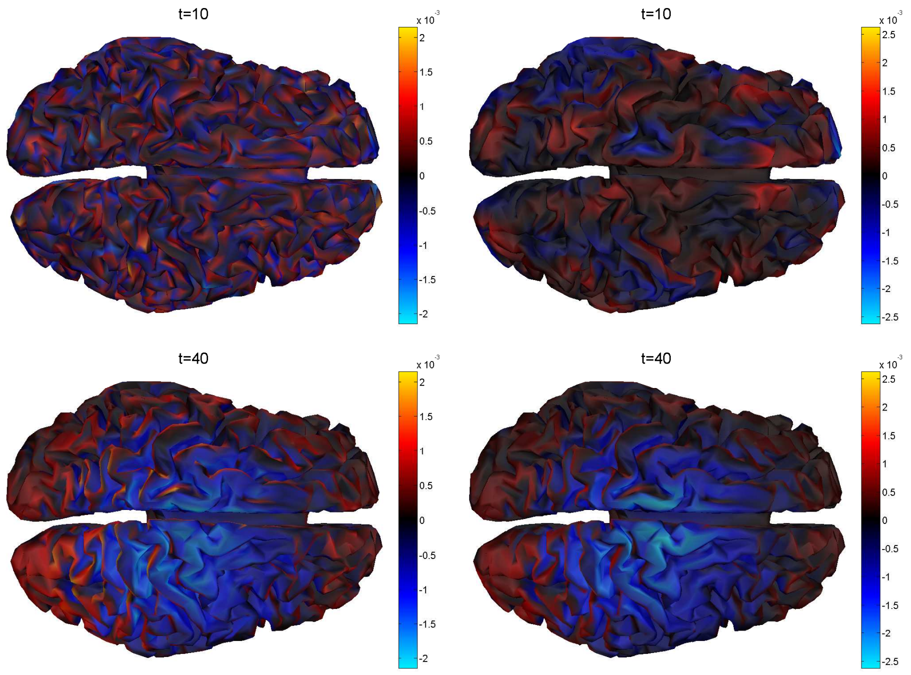

Figure 5.11: *On the left the true source is plotted on the cortical surface at two different time indices (10ms and 40ms) and on the right the estimated source is plotted for comparison.*

5.1.4 Discussion

The simulations presented using Algorithm I have revealed some basic properties of the algorithm. Lets summarize some of the main points here. For well posed square systems, where prior and likelihood distributions are accurately assumed Gaussian distributed, the algorithm performed perfectly. Variance estimates from hyperparameters were very good within a a lower limit boundary of of the hyperparameters. This boundary does not cause any practical problems. For ill-posed systems the noise estimate deviates a bit without causing too much errors, in the sense that the algorithm still converges and produces useful results. Even for very non-Gaussian source distributions Algorithm I estimates are usable and performance is far better than the method of least squares.

5.2 Algorithm II

Algorithm I is simple, easy to implement and converges fast. But for very sparse and/or localized sources its performance decreases since the source distribution is not Gaussian. Algorithm II was based on the ARD formulation and should in theory be able to better detect sparse sources. There are however some issues with Algorithm II. The number of parameters to be estimated is much larger than in Algorithm I making its computation heavier and convergence therefore slower. Here some examples and evaluations of Algorithm II will be presented. These examples justify the expansions leading to Algorithms IIb, IIc and IId.

5.2.1 Evaluation of Algorithm II

The hyperparameters returned by Algorithm II should give the noise variance and indication of relevance for each source. More precisely according to equation 4.61 the β value should give the inverse of the noise standard deviation, as was also the case in Algorithm I. Equation 4.61 shows that each α_k gives the relevance of its corresponding source s_k, i.e. if the source is zero α_k should approach infinity and if the source is active (non-zero) α_k should be a finite number inversely proportional to the source magnitude. Figure 5.12 shows graphs

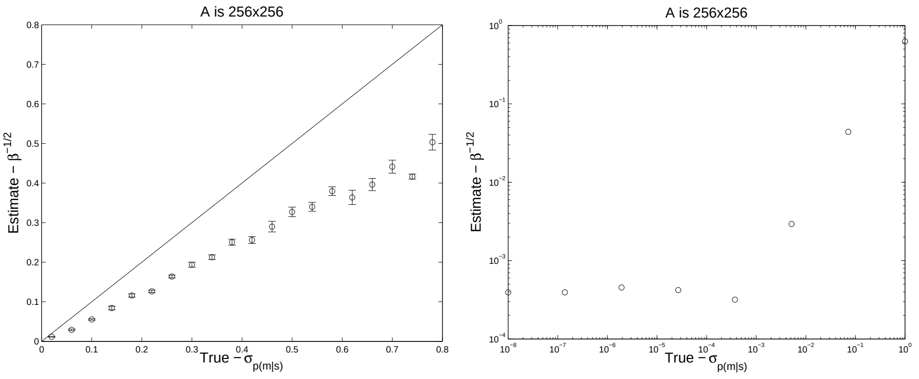

Figure 5.12: *Comparison of estimated and true noise variance when running Algorithm II on a well-posed system of dimensions $N = P = 256$. Each estimate is an average over 10 runs with the corresponding error bars. The source variance is kept constant for all noise levels giving SNR ratio from approximately -10dB to 22dB. The elements of the* **A** *matrix are drawn from a Gaussian distribution. Ideally both graphs should be straight lines with unity slope. For low noise values a saturation is present as the graph on the right shows.*

of estimated vs. true noise for Algorithm II for a well posed square system ($N = P = 256$). As the graph on the left shows there is an underestimation of the noise which can result in overfitting. The graph to the right on the figure shows saturation for low noise values as was also present in Algorithm I. Underestimation of noise becomes even worse as the system becomes more ill-posed and gradually as P becomes larger than N the β estimate starts to approach infinity so the only limiting factor on the magnitude of β is the number of iterations.

This larger underestimation of noise and non-convergence of β for more ill-posed systems was not as apparent in Algorithm I and motivated the construction of Algorithm IIb, where the noise parameter β is first estimated using Algorithm I and then the update equations for α_k ($k = 1, ..., P$) and $\mathbf{s}_{MP}$ from Algorithm II are used. This gives a better estimate of the noise variance reducing the possibility of overfitting. Figure 5.13 shows a simple test using Algorithm IIb on an ill-posed system with $N = 256$ and $P = 10201$ and the $\mathbf{A}$ matrix created as before by drawing numbers from a normal distribution. The dimensions here

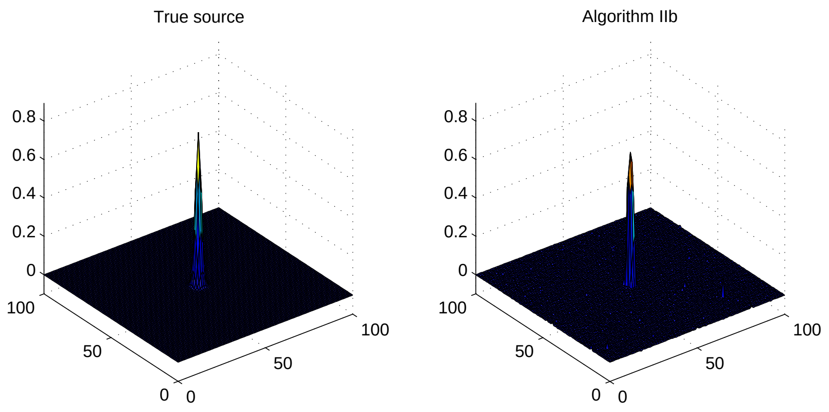

Figure 5.13: *True artificial source and the estimated source using Algorithm IIb on a system with $N = 256$ and $P = 10201$. The SNR is set to 10dB and $\mathbf{A}$ is created by drawing numbers from a normal distribution with standard deviation of 0.1. Note that axis labeling has been omitted here. It should be clear that the x- and y-axis represent the source space locations and the z-axis represents the source magnitude. This will also be the case on similar figures presented later in this chapter.*

are similar to the ones in the more realistic lead-field matrices, like the 3 sphere used in Algorithm I simulations and the BEM models which will be used later on. The only difference lies in the P dimension. For these simple 2D simulations the square root of P is preferred to be a natural number so that a discrete $\sqrt{P} \times \sqrt{P}$ source grid can be formed. The true source on the left of figure 5.13 is a pulse on 2D plane, similar to one of the toy examples shown for Algorithm I. On the right the estimated source from Algorithm IIb is shown and it closely resembles the true source. Furthermore on figure 5.14 the convergence of some parameters is shown. Two of the overall 10201 α_k values are shown, α_2 is converging to infinity and α_2 to a constant value, i.e. irrelevant and relevant classes respectively. α_1 is the smallest α_k value and its corresponding source is the one with the highest amplitude on figure 5.13. Finally it is worth noting that the noise variance in this simulation is 4×10^{-3} and estimated noise variance from the algorithm is slightly underestimated at 2×10^{-3}.

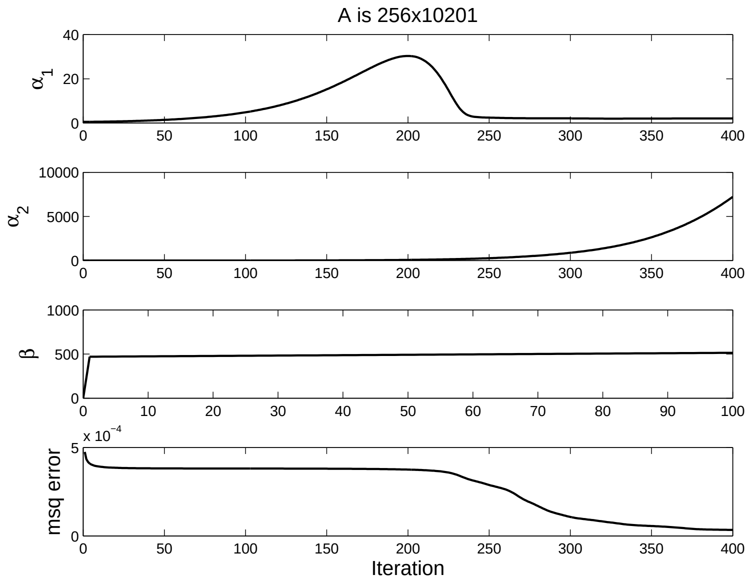

Figure 5.14: *Convergence of Algorithm IIb, β is iterated 100 times using Algorithm I and then the other parameters are iterated 400 times. Only two of the total of 10201 α_k values are shown, one converges toward infinity, irrelevant class, and the other converges to a constant, relevant class. Also shown is the mean squared error over iterations (msq error).*

5.2.2 Inspection of Active Sets

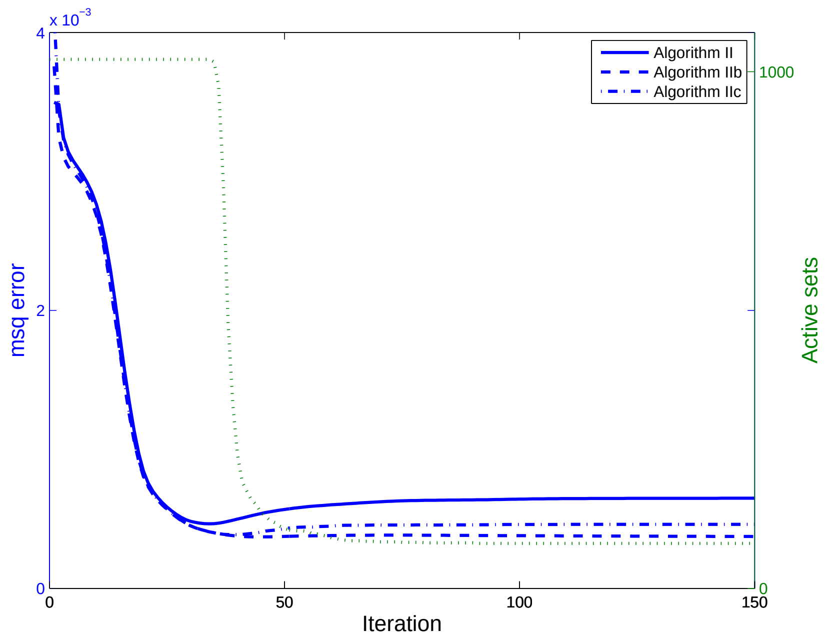

Figure 5.15: *Mean squared error plotted as a function of iteration when running Algorithms II, IIb and IIc (blue curves). Also plotted on the graph is the number of active sets over iterations for Algorithm IIc, shown as a dotted green curve with y-axis to the right in the graph. The system is ill-posed with* $N = 256$ *and* $P = 1024$*. Number of iterations are 150 for all algorithms except when estimating* β *in IIb and IIc then 100 iterations are used when running Algorithm I.*

In section 4.2.6 it was introduced how inactive sets could be removed from the model during the iteration of Algorithm IIb. This requires the definition of an α_{k-max} value, so that when α_k exceeds this value it is removed from the model along with its corresponding source. This threshold value is crucial, too low and relevant sources are removed from the model and too high no sources are removed from the model.

On figure 5.15 the mean squared error is plotted as a function of iteration when running Algorithms II, IIb and IIc. Also plotted on the graph is the number of active sets over iterations for Algorithm IIc (green curve) with y-axis labeled to the right. The system in this example is ill-posed with $N = 256$ and $P = 1024$, smaller dimension for P is used here than was used in last section to speed up calculations since we are evaluating the effect of removing active sets. Maximum α_k value is chosen $\alpha_{k-max} = 10\alpha_{algI}$, where α_{algI} is the α value returned by Algorithm I. On the graph it can be seen how the number of active sets decreases steeply just as the algorithm is about to converge speeding up the final calculations. More interestingly the graph also shows the effect of bad noise estimation. For ill-posed systems we remember that Algorithm II was

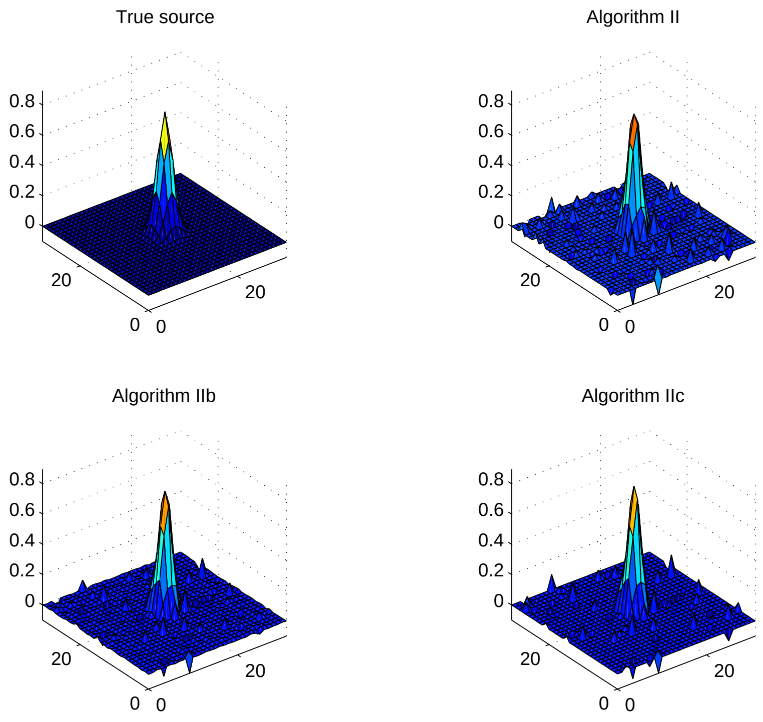

Figure 5.16: *True source and source estimates for the example presented in figure 5.15.*

prone to making a bad estimate of β, and sometimes even blowing up toward infinity. This was the main reason for formulating Algorithm IIb which, along with Algorithm IIc, uses Algorithm I to estimate the noise. In the example the true noise variance is $\sigma_\epsilon^2 = 4.04 \times 10^{-3}$ and the estimated noise variances from the three algorithms are listed in table 5.1. The noise estimate for Algorithm II is far worse than for IIb and IIc as expected. On the graph on figure 5.15 it can be seen how this causes the mean squared error to reach a minimum value and then increase again before converging, i.e. there is some overfitting for Algorithm II. There is also a slight overfitting for IIb and IIc but not as much because the overestimation of the noise is far less. The true source and source

Algorithm	**Estimated σ_ϵ^2**
II	4.59×10^{-5}
IIb	3.01×10^{-3}
IIc	2.96×10^{-3}

Table 5.1: *Noise estimates returned by the three variants of Algorithm II.*

estimates can be seen on figure 5.16. All algorithms return reasonable estimates and the visual difference is not that great between the estimates.

Figure 5.17 shows the effect of having too low α_{k-max} value. Its value is 100 times smaller than before which causes the early pruning of the model so the algorithm hits a plateau before reaching the minimum error possible. Finally we can conclude this section by noting that if it is guaranteed that α_{k-max} is not too low than as a major reduction in active sets becomes present the algorithm has converged. Therefore if we could find some good criterion for choosing α_{k-max} the number of active sets could be used as a convergence indicator.

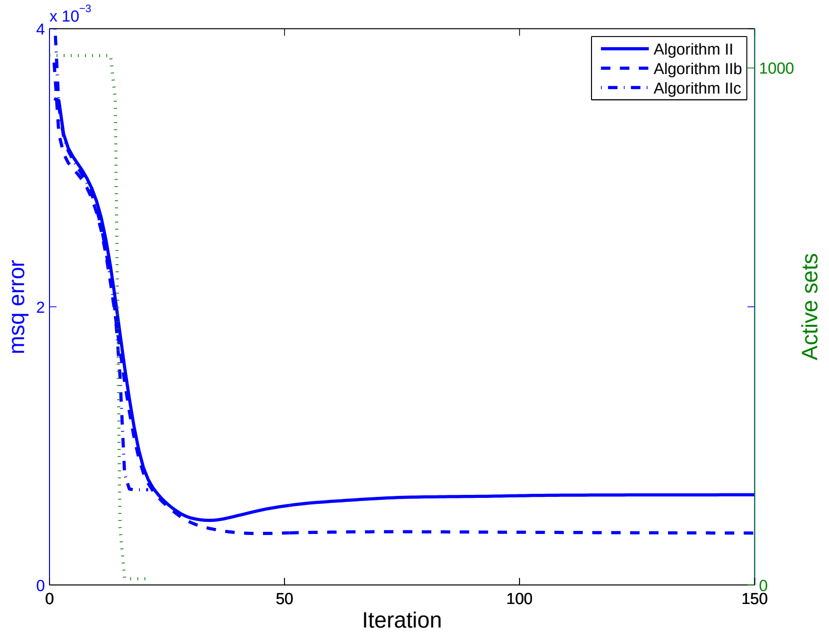

Figure 5.17: *Same system as in previous example but the α_{k-max} value is 100 times lower than before causing early pruning of the model. Algorithm IIc therefore stops iterating before reaching the minimum error.*

5.2.3 Ill-conditioned BEM Head Model

The examples so far have showed that the ARD algorithms work even for very ill-posed systems. But when doing more realistic simulations using a 3 sphere head model or a BEM head model the condition number of the lead field matrix $\mathbf{A}$ becomes much higher. Section 3.4 gave an introduction on BEM head models and in section 3.5 the software package used to calculate the head models was described. The same BEM head model is used in all BEM sections of this chapter. For the sake of reproducibility all details regarding this model are listed in appendix C.1. The lead field matrix $\mathbf{A}$ for the BEM head model has dimensions $N = 256$ and $P = 10001$. Its condition number is $\kappa = 5.93 \times 10^8$. Its singular

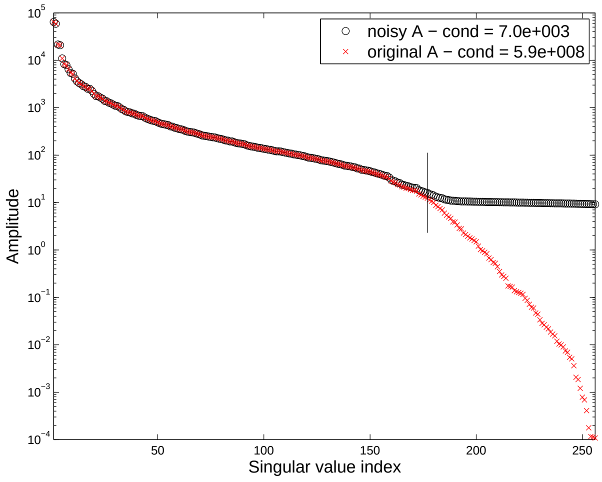

Figure 5.18: *Singular values of lead field matrix* $\mathbf{A}$ *for a BEM head model plotted with x's. Also plotted are the singular values after adding Gaussian noise to* $\mathbf{A}$ *from a distribution with zero mean and standard deviation of 0.1.*

values are plotted on figure 5.18 showing how they span eight orders of magnitude. It is important here to note that in the examples presented in preceding sections where the matrix $\mathbf{A}$ was created by drawing numbers from a normal distribution the condition number is much less, e.g. for a random $\mathbf{A}$ of size 256×10001 with standard deviation 0.1 the condition number is around 1.38, i.e. zero orders of magnitude compared to eight for the BEM model. The extremely high condition number for the BEM model causes the ARD algorithms to give very poor results, rendering them useless in most cases.

Different tricks can be used to try and increase the condition number of the model. The first one is adding some noise to $\mathbf{A}$, the effect of doing so can be seen on figure 5.18 where the singular values of $\mathbf{A}$ are plotted before and after adding Gaussian noise drawn from a distribution with standard deviation of 0.1. Doing so increases the condition number significantly by five orders of magnitude but even so the results from the ARD algorithms are poor (even if the noisy matrix is also used for the forward calculations). Another trick is removing all singular values that are below a certain threshold from the model, e.g. removing all values to the right of the vertical line on figure 5.18 gives a system with condition number of same magnitude as was achieved by adding the noise. But as was the case with the noisy $\mathbf{A}$ the results are also poor when removing singular values.

Since systems with condition numbers of magnitudes around 3 do not give good results when using the ARD algorithms it is of interest to try and make a test which gives an indication of how high the condition number can be so that the ARD algorithms give satisfying results. This can be done by generating arbitrary matrices $\mathbf{A}$ with condition numbers ranging from 0 to 8 order of magnitude. Let $\mathbf{A}_0$ be the BEM lead field matrix ($\kappa = 5.93 \times 10^8$) and $\mathbf{A}_n$ be the matrix generated by drawing numbers from a normal distribution ($\kappa = 1.38$). By mixing these two matrices a matrix $\mathbf{A}$ with a condition number ranging from 0 to 8 order of magnitude can be generated. E.g. the equation

$$\mathbf{A} = q\mathbf{A}_n + (1-q)\mathbf{A}_0$$

where $0 \leq q \leq 1$ can be used to vary the condition number. Running tests for systems of different condition number indicated that when the condition number has order of magnitude 2 or higher the results become too poor to be considered useful.

In general one would expect that the activity over the cortical surface to be smooth, meaning that if a source is active then its neighboring sources are more likely also to be active than other sources placed further away. The artificial sources used so far in the simulations comply with this assumption. This leads to an intuitive method, which was presented in section 4.2.7, to try and improve the performance of the ARD algorithms. The method incorporates spatial smoothing (low pass filtering) of the sources into the iteration procedure of Algorithm IIb. Algorithm IId refers to Algorithm IIb with this smoothing feature added. On figure 5.19 results are illustrated when this method is used on the BEM system above above ($\kappa = 5.93 \times 10^8$). Two values of q are tested, $q = 0.4$ and $q = 0.5$, and the algorithm calculates 3000 iterations. SNR in this example is 10.3dB. The true source is on the bottom in the figure and the estimates on the top. The location of the estimate is good but the cortical size is a bit small. There is also a bit high amplitude in the estimates approx. 2 to 3 times higher than the true amplitude, but overall the estimates are fairly good and incorrect artifacts are so small that they are not visible on the figure. Finally we note that the true noise variance is 2.4×10^{-9} and the estimated noise variance for

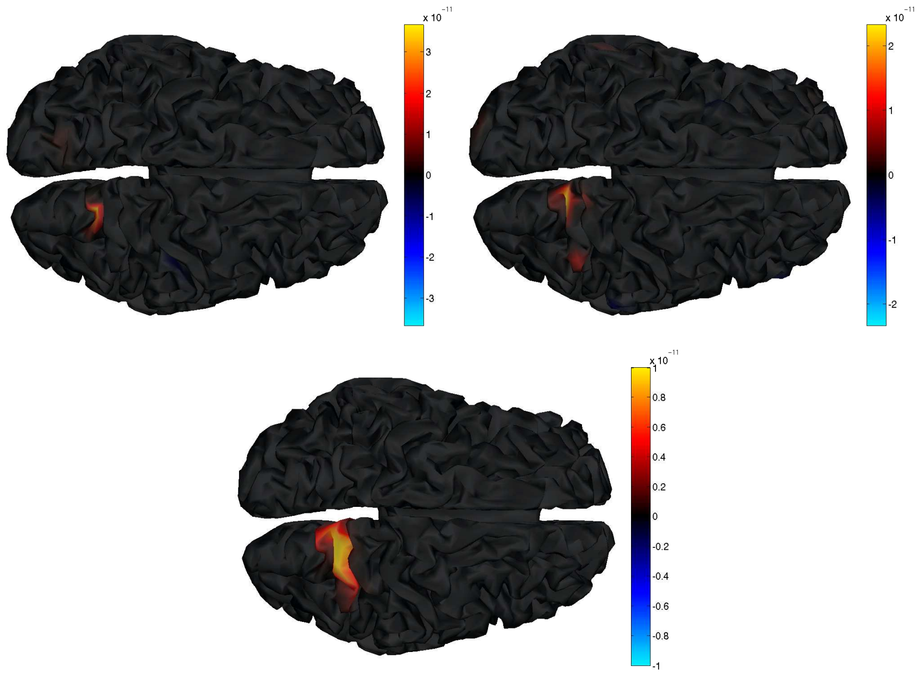

Figure 5.19: *Running Algorithm IId on a very ill-conditioned BEM head model. True source is on the bottom figure and the algorithm estimate can be seen on the two top figures for different q values (eq. 4.72). Namely the left figure has $q = 0.4$ and the right figure has $q = 0.5$.*

this example is 2.3×10^{-9}, very close to the true value.

The factor q in the smoothing of Algorithm IId leads to the obvious task of finding the optimum value of q, note that $q = 1$ gives Algorithm IIb. This depends on the system properties, e.g. in our BEM model the number of nearest neighbours for each source is varying and the average number of neighbours is 6. But for the 2D examples the number of neighbours 4 and constant. As has also been noted the quality of the noise estimate influences how good the source estimate is. Assuming good noise estimate we would expect that the dimensions $N \times P$ and the number of neighbours would have the biggest influence on the optimum q. Figure 5.20 shows that $q = 0.4$ is the optimum value for a 2D example system of dimensions 256×1024 and for a more ill-posed system of dimensions 256×10201 the optimum q value is slightly higher. For the ill-conditioned BEM example the same graph is plotted on figure 5.21 (see also figure 5.19 for more details). The msq error improves as q decreases an the graph indicates an optimum value of 0. The change in error is however very small below $q = 0.6$ and

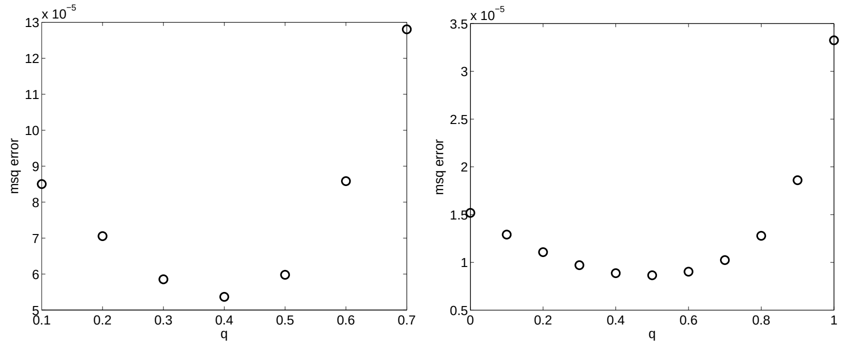

Figure 5.20: *Mean square error for Algorithm IId as a function of q for a simple 2D surface example with dimensions* 256×1024 *on the left graph and* 256×10201 *on the right graph.* $q = 0.4$ *is the approximate optimum value for the smaller system and* $q = 0.5$ *for the larger. SNR=10dB and iterations for* β *are 100, for the other parameters iterations are 200 and 400 respectively. Noise standard deviation estimates on the left are around* 5.5×10^{-2} *for all q values and the true noise standard deviation is* 6.4×10^{-2}*. On the right noise estimates are around* 2.1×10^{-2} *and true noise is* 6.5×10^{-2}*. Matrix* $\mathbf{A}$ *is created as before.*

the ill-conditioned $\mathbf{A}$ most likely causes the error not to rise when we reach zero. We therefore choose q usually between 0.4 and 0.5 in following sections where Algorithm IId is used.

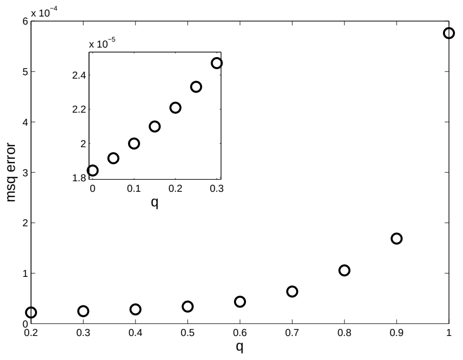

Figure 5.21: *Mean square error for Algorithm IId as a function of q for on the ill-conditioned BEM system (see also figure 5.19). There is a definite improvement as q becomes smaller, and below 0.6 the msq error seems to converge. The minimum is however in* $q = 0$*, but for obvious reasons we can not choose that value and assume that the condition number of* $\mathbf{A}$ *could be the cause for the error not to rise for* $q = 0$*.*

5.2.4 Effect of SNR

In the simulations so far the SNR has been fixed and usually in the range from 5 to 10dB. SNR plays a big role in the expected performance of the algorithms and here the plan is to check the relationship between SNR and the msq error for the ARD algorithms.

A simulation is done using **A** of dimensions 128 $\times$ 1024 where the matrix elements are drawn from a normal distribution of standard deviation 0.1. ARD Algorithm IIc with 400 iterations is used for faster calculations with α_{k-max} high enough so that the algorithm fully converges (see section 5.2.2 for more details on Algorithm IIc). The mean square error value of the source estimates is used as a measure of the quality of the estimate. Figure 5.22 shows a graph

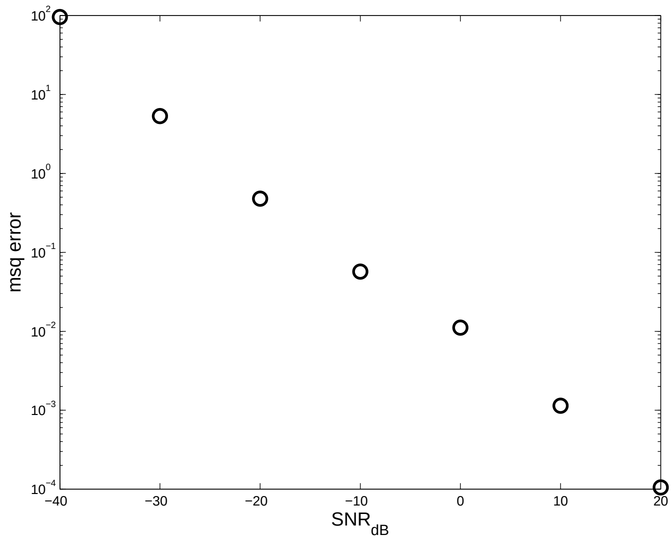

Figure 5.22: *Mean square error of source estimates as a function of SNR for a system of dimensions* 128 $\times$ 1024. *A linear relationship is clearly present.*

of the mean squared error (msq error) as a function of SNR indicating a linear relationship between the two, i.e. the estimate from the ARD algorithms is directly proportional to the SNR. Figure 5.23 shows four plots of source estimates for different SNR values. For SNR = 20dB the estimate is practically the same as the true source, for SNR = 0dB the true source structure is visible in the source estimate but there are large artifacts present. Below SNR = 0dB the source estimates are very poor and far from providing useful information.

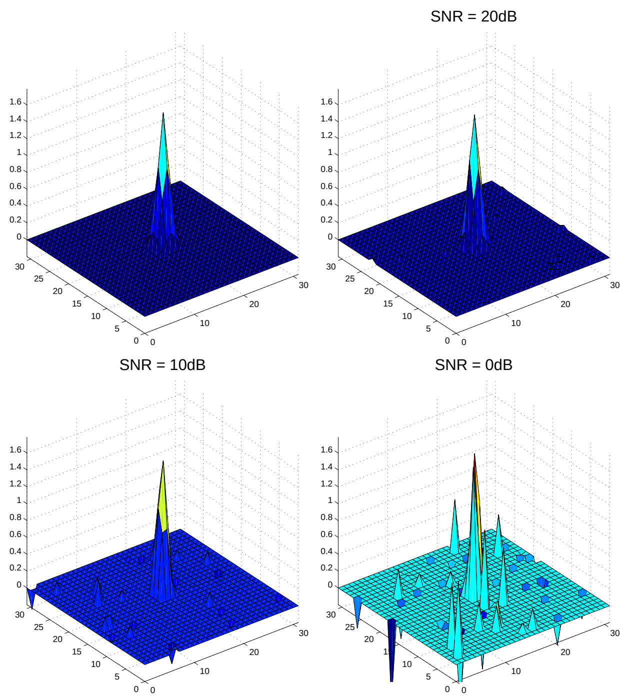

Figure 5.23: *Source estimates for different SNR values showing how much the SNR ratio effects the results. The true source is shown in the top left corner, estimate for SNR=20dB is practically equal to the true source.*

5.2.5 Discussion

Simulations for Algorithm II have now been presented. The sources used were specially suited for the ARD prior in the sense that they were sparse with few active sources. Algorithm II therefore naturally outperformed Algorithm I for these examples, although not specially shown here. Improvements of Algorithm II were justified with the examples presented. It was shown how the noise estimate from Algorithm II was worse than the noise estimate from Algorithm I, thus Algorithm IIb uses the noise estimate from Algorithm I. Since the irrelevant hyperparameters approach infinity when running Algorithm IIb (and II also) the inactive sets were removed during iteration in Algorithm IIc. But one has to be careful when choosing the removal threshold, too low threshold degrades the performance. It was also shown how the noise estimate influences the quality of the source estimate by comparing the msq error curves for Algorithms II, IIb and IIc. Algorithm II msq error increased after reaching a minimum error since its noise estimate was worse than for Algorithms IIb and IIc For realistic ill-conditioned **A** matrices spatial smoothing had to be added to the iteration loop to get any useful results. Algorithm IId was therefore defined by adding a low pass filter to Algorithm IIb.

5.3 Algorithm III

Algorithm III is not based on as solid theoretical framework as Algorithms I and II. One can say that it is constructed from a mixture of two different prior assumptions. The priors are however closely related and one would expect that the estimates from Algorithm III to be very similar to the estimates from Algorithm IId. In this section Algorithm III will be tested and more importantly some tests will be done comparing all of the algorithms.

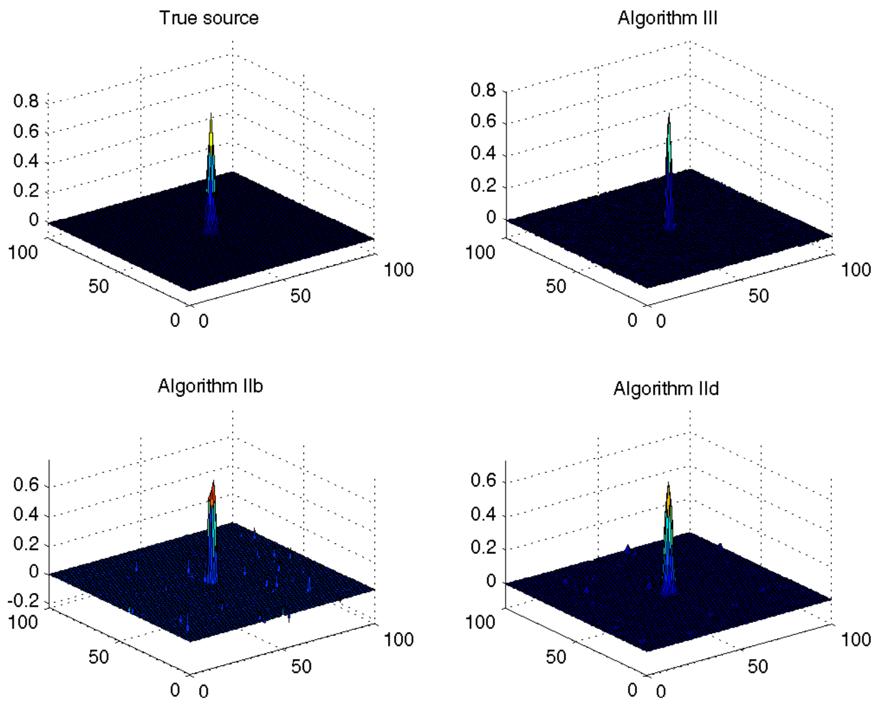

Figure 5.24: *True source and the estimated source using Algorithms IIb, IId* ($q = 0.4$) *and III on a system with* $N = 256$ *and* $P = 10201$. *The SNR is set to 10dB and* **A** *is created as in previous examples.*

5.3.1 Evaluation of Algorithm III

The β hyperparameter returned by Algorithm III is estimated using Algorithm I. We expect hyperparameters α_k to convergence in a similar way as in Algorithm II. If source s_k is zero α_k should approach infinity and if the source is active (non-zero) α_k should be a finite number inversely proportional to the source magnitude.

Figure 5.24 shows a simple test using Algorithm III along with Algorithms IIb and IId ($q = 0.4$) on an ill-posed system with $N = 256$ and $P = 10201$ and the $\mathbf{A}$ matrix created as before by drawing numbers from a Normal distribution. The true source and estimates are labeled respectively on the figure. All the source estimates closely resemble the true source but the low pass filtering effect in Algorithm III is a bit strong reducing the area of the source. There are more disturbances in the Algorithm IIb estimate than in the Algorithm IId estimate as expected. The best estimate is from Algorithm IId. Noise variance

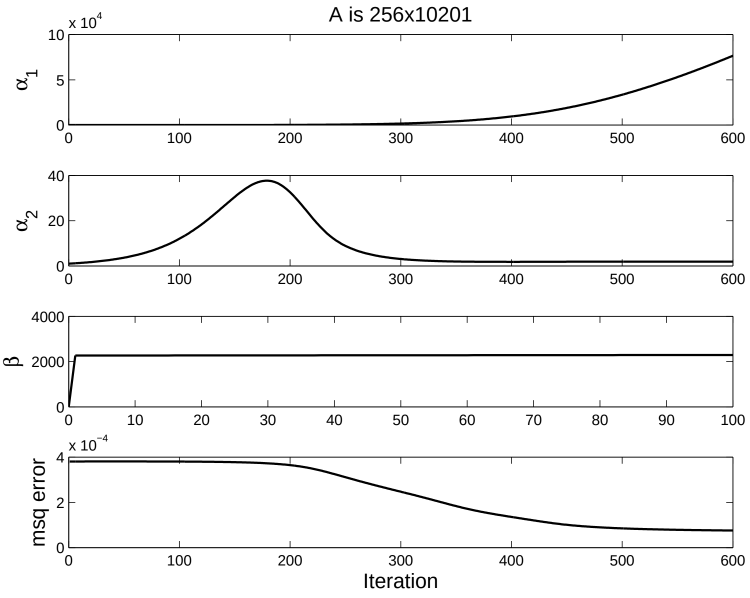

Figure 5.25: *Convergence of Algorithm III, β is iterated 100 times using Algorithm I and then the other parameters are iterated 600 times. Only two of the total of 10201 α_k values are shown, one converges toward infinity, irrelevant source, and the other converges to a constant, relevant source. Also shown is the mean squared error over iterations (msq error).*

estimates are in all cases 4.4×10^{-4} and very close to the true noise variance of 4.2×10^{-4}. Finally on figure 5.25 the convergence of some parameters is shown for Algorithm III. Two of the overall 10201 α_k values are shown, α_1 is converging to infinity and α_2 to a constant value, i.e. irrelevant and relevant sources respectively. α_2 is the smallest α_k value and its corresponding source is the one with the highest amplitude of the Algorithm III estimate on figure 5.13.

Algorithm III works as was expected, i.e. α_k values are inversely proportional to their corresponding sources like in Algorithm II, IIb, IIc and IId. The low pass filtering effect is stronger than in Algorithm IId.

5.3.2 SNR Comparison for Algorithms

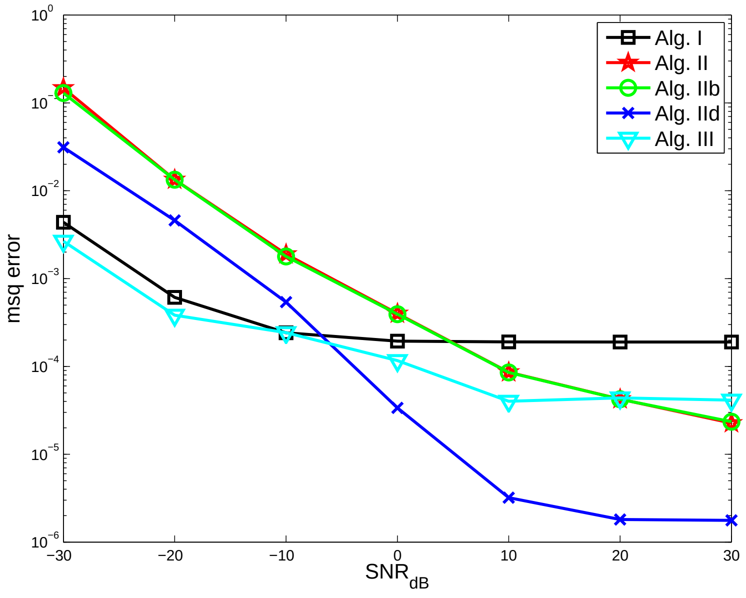

Figure 5.26: *Mean square error of source estimates as a function of SNR for a system of dimensions* 256×10201*. All algorithms are compared here except Algorithm IIc because it should give same results as Algorithm IIb, only with reduced computation time. In Algorithm IId we choose* $q = 0.4$ *and iterations are set to 100 (Alg. I) and 500 in all runs. The overall best performance is from Algorithm IId since below -10dB no algorithm gives a usable source estimate.*

A simulation is done using $\mathbf{A}$ of dimensions 256×10201 where the matrix elements are drawn from a normal distribution of standard deviation 0.1. Figure

5.26 shows a plot of the mean squared error as a function of SNR for Algorithms I, II, IIb, IId and III. Algorithm IIc is not shown because the only effective difference between IIc and IIb is computation time. Below SNR of -10dB Algorithm III and Algorithm I perform best but unfortunately the estimates from all algorithms are useless for that low SNR. By useless we mean that there is no detectable pulse at the right location in the estimate. On the practical range above -10dB Algorithm IId performs best and we therefore say that for this example it is the best choice. By initializing the hyperparameters to 1 in Algorithm I the noise estimates are the same for all algorithms except Algorithm IIb. The effect of different noise estimates is therefore minimum.

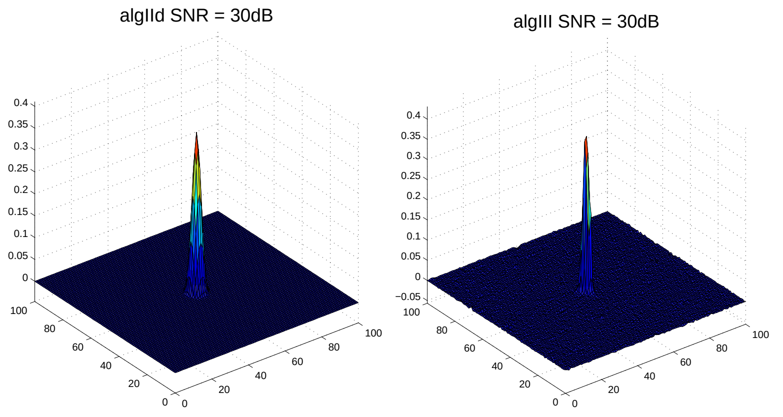

Figure 5.27: *Estimates from Algorithms IId and III for SNR=30dB. The IId estimate is practically equal to the true source.*

The source used in the test can be seen from the estimate of Algorithm IId on figure 5.27. The poor performance of Algorithm I is understandable when the source is inspected. This kind of sparse localized source is tailored to the ARD and smoothing prior assumptions. Figures 5.28 and 5.29 show plots of source estimates for SNR values of 0dB and 10dB. We can confidently conclude that the overall best performance is from Algorithm IId but Algorithm III is better than I, II, IIb and IIc. Algorithms IId and III give useful estimates down to 0dB, contrary to the others.

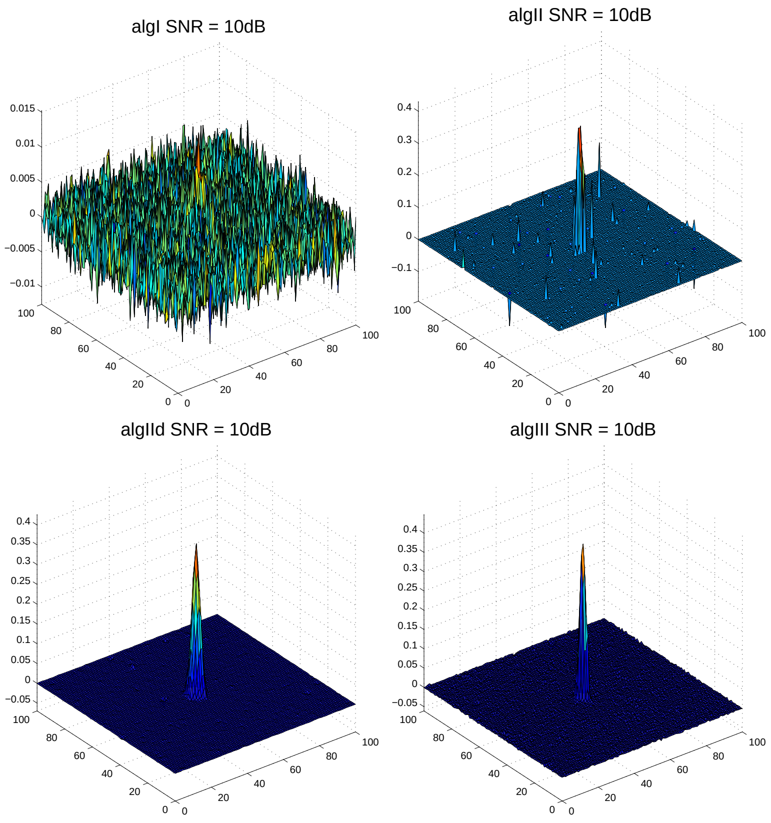

Figure 5.28: *Source estimates for SNR value of 10dB. Algorithm IIb is not shown since the graph on figure 5.26 indicates that II and IIb have practically the same performance. The worst performance is from Algorithm I, but looking closely at the figure it can be seen that there is an increased activity in the estimate where the true pulse should be located.*

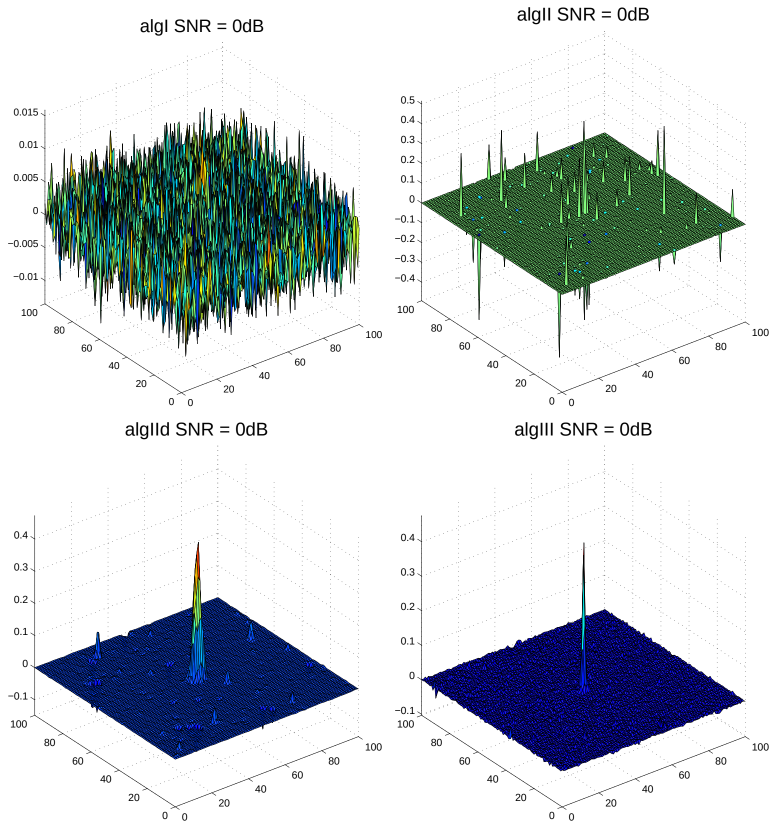

Figure 5.29: *Source estimates for SNR value of 0dB. The superior performance of Algorithms IId and III can be seen here.*

5.3.3 SNR Comparison with BEM

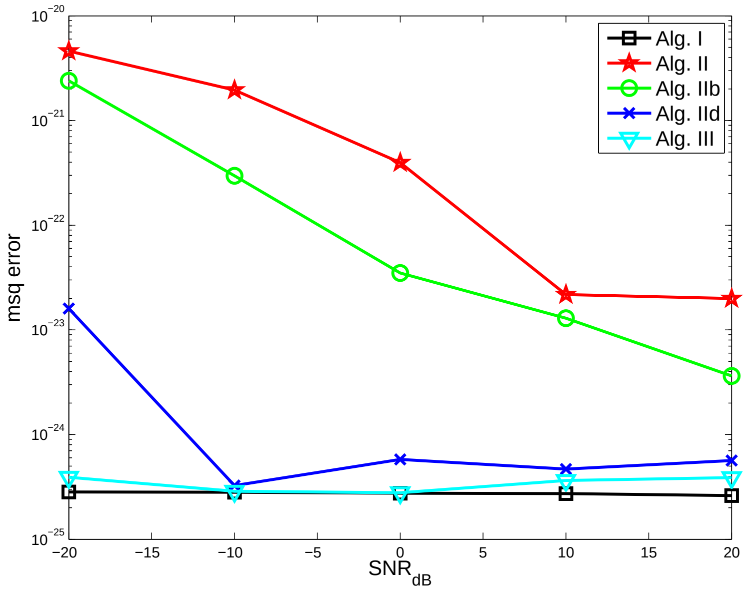

Figure 5.30: *Mean square error as a function of SNR for Algorithm I, II, IIb, IId and III. The forward model is an ill-conditioned BEM model of dimensions* 256×10001*.*

In section 5.2.3 there was a discussion on Algorithms II, IIb, IIc and IId when running them using ill-conditioned BEM head models. The overall conclusion from that section was that of all the ARD algorithms Algorithm IId was the only one that gave good estimates. Algorithm III is an attempt to make a more general implementation of Algorithm IId so comparing the two is of interest. Here the same system and setup parameters (3000 iterations and $q = 0.4$ in Alg. IId) will be used as in section 5.2.3 to test and compare all algorithms on an ill-conditioned BEM head model. Algorithm I has a stop threshold of 10^{-3} and max iterations of 100. Figure 5.30 shows the estimate mean square error as a function of SNR. Algorithm IIc was skipped from this test since the only difference between Algorithm IIb and IIc is computation time. As previous examples have indicated Algorithms II and IIb have poor performance when using realistic forward models. Algorithm I has the best performance here with Algorithms IId and III following closely. In table 5.2 the true noise variance is compared with the estimated noise variance. The noise estimates from Algorithm I are better than noise estimates from Algorithm II in all cases except for SNR $= -10$dB. It can also be noted that the noise estimates are in all cases good.

SNR	**True noise σ_ϵ^2**	**Estimated noise $\frac{1}{\beta}$**	
		Alg. I, IIb, IId, III	Alg. II
-20dB	5.9×10^{-15}	5.9×10^{-15}	4.8×10^{-15}
-10dB	5.9×10^{-16}	6.6×10^{-16}	5.3×10^{-16}
0dB	5.9×10^{-17}	5.5×10^{-17}	3.9×10^{-17}
10dB	5.9×10^{-18}	6.0×10^{-18}	4.5×10^{-18}
20dB	5.9×10^{-19}	5.6×10^{-19}	4.0×10^{-19}

Table 5.2: *True noise variance and estimated noise variance for the tests shown on figure 5.30. As expected the overall noise estimates from Algorithm II are worse.*

The true source used here, visualized on figure 5.31 is similar to the one from section 5.3.2, i.e. a pulse with zero background activity. The only major difference between the tests here and in section 5.3.2 is the condition number of the lead-field matrix **A**. This causes huge effect on the algorithms performances and seriously degrades the estimates from all except Algorithm I. In previous section Algorithm I had the worst overall performance but here it is the best. The estimates from Algorithm IId and III are not far from the estimates from Algorithm I but the much larger computational overhead of Algorithms IId and III is hardly justifiable when their performance does not exceed Algorithm I, which is much simpler and more robust. On figure 5.31 the estimates from all algorithms at SNR = 0dB are plotted. These plots show how bad the estimates from Algorithm II and IIb are. Estimates from Algorithm I, IId and III however good.

5.3.4 Discussion

Evaluation of Algorithm III showed that it gives very similar results as Algorithm IId but the performance of Algorithm IId is slightly better in general. The computation of Algorithm III is a little bit more complicated than Algorithm IId, one would therefore choose Algorithm IId over Algorithm III.

All algorithms were compared over a range of SNR both for a well-conditioned and a realistic ill-conditioned lead-field matrix. The sources used were sparse localized pulses with no background activity. These kind of sources should be less suitable for Algorithm I. For the well-conditioned lead-field matrix Algorithm IId had the best performance. But for the more realistic lead-field matrix Algorithm I was the best.

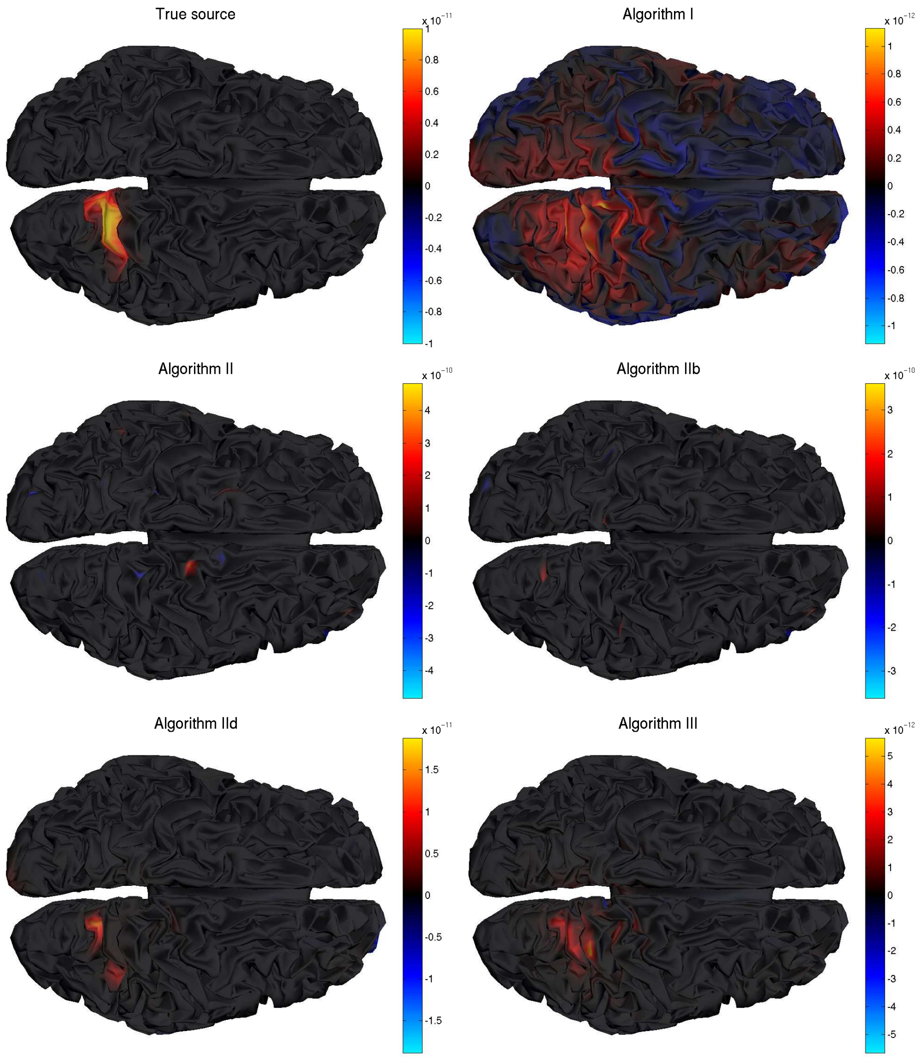

Figure 5.31: *True source on the top left figure and algorithm estimates for SNR of 0dB. Estimates of the different algorithms are labeled.*

Chapter 6

Real Data Testing

After checking the performance of the algorithms in last chapter it is of interest to try them on real EEG data recorded from a subjects scalp. Here real data will be analyzed and inverse solutions presented. The dataset used is available free for research purposes. In appendix D analysis of some sample data from BrainStorm is also presented. It consists of 256 channel EEG recordings over a 6s time period. Very little information on this data is supplied making the validation of inverse solutions hard. Due to the lack of information this analysis is presented in an appendix. It is worth noting that some nice inverse solutions using Algorithms I and IId were obtained on that data. The estimates from both algorithms complied in the sense that the same areas were active. The only major difference was the more sparse estimate from Algorithm IId.

6.1 BCI Competition III Data

The data to be analyzed is from the *BCI Competition III* [34], a competition held to validate signal processing and classification methods for *Brain-Computer Interfaces (BCIs)*. This competition is from 2004 and all the data from it is available for research purposes. The datasets were divided into five different classes where different practical BCI problems were addressed. Here dataset

IVa [35] from the competition will be use. Since the focus of this thesis is on the EEG inverse problem the dataset is not used in the way it was intended for the competition, namely classification, and only a small fraction of it will be analyzed. The data was recorded from five healthy subjects sitting in a comfortable chair with arms resting on armrests. Visual cues were indicated for 3.5s and during them the subject should perform a *motor imagery*. The process of visualizing motor executions is known as motor imagery. Three possible imageries presented were, left hand, right hand and right foot. Between presentations of target cues the subject could relax for 1.75 to 2.25s. In the competition data cues were only provided for right hand and right foot, so in this analysis the cue is a motor imagery of a right hand or foot. This is important for reasons that will become clear soon. Recordings were made using a 128 channel electrode cap from Electro-Cap International Inc. (ECI) and 118 of them were measured. The data is band-pass filtered between 0.05 and 200Hz and digitized at 1kHz. Here a 100Hz down sampled version of the data is used.

6.2 Method

When analyzing real EEG data inverse solutions one has to be very careful. The ill-posed nature of the problem which has been discussed many times so far means that the number of possible solutions is very large. The large condition number of the lead field matrix for realistic head shapes causes numerical problems and the quality of the tesselation surfaces for the subject must influence the solution quality. The study here will be based on a paper by *Qin et al.* [36]. They performed motor imagery classification using source analysis on a data set where following a resting period motor imagery cues were presented, i.e. a very similar data set as the one used here. In the study the focus was on so called *mu* rhythm which is caused by movement and also by motor imagery. This rhythm is part of the alpha band, recorded over the sensory motor cortex, that decreases or desynchronizes with movement and motor imagery. It is therefore expected that at the start of a motor imagery there will be an event related decrease in the mu rhythm in the brain region controlling the movement. Note that the left side of the body is controlled by the right side of the brain and vice versa. In our case the motor imagery is on the right side of the body meaning the left motor cortex mu rhythm should be blocked but activity on the right motor cortex should be anticipated.

The basic idea is to use frequency analysis to locate a decrease in the mu rhythm of the data, which should happen at the start of the motor imagery. Having located at which time the decrease happens we will solve the inverse problem around the selected time. There we anticipate activity on the right side of the motor cortex, i.e. the area not controlling the motor imagery. Since the motor imagery cues are visual we might also expect some activity in the visual cortex, which is located on the back side of the brain.

6.3 Subject Forward Model

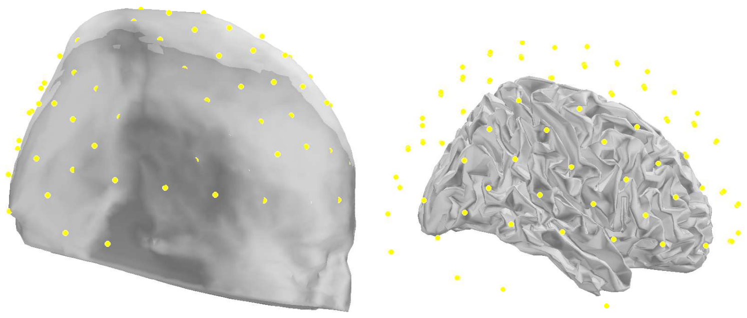

Figure 6.1: *Channels and two of the head tesselation surfaces used for the BCI data BEM calculations. On the left the channels are shown with respect to the scalp, there are some minor intersections of channels and scalp causing some channel locations to be slightly inside the scalp. The right figure shows the brain tesselation surface and channels.*

Since there was no subject head information with the data set the Montreal Brain Phantom warping procedure from section 3.5 was used. But to do that the 3D locations of the 118 electrodes are needed. Unfortunately the channel locations were only given in 2D coordinates, like illustrated on figure 2.3 (page 12) for the 10-20 system. This kind of 2D map is not subject dependent and only indicates the relative placements of electrodes to each other. This means that we have no information on the subjects head structure. One could therefore argue that using a spherical head model here would be justifiable, but to avoid leaving out parts of the brain a BEM head model will be used. A 3D channel template is therefore needed, a kind of a ”channel location phantom”. *Oostenveld* and *Praamstra* [17] have suggested an expansion of the 10-20 EEG system as described in section 2.2. On Oostenveld’s webpage [18] he supplies a computation of electrode locations on a realistic head surface. Using this template the 2D channel locations supplied with the data could be matched with Oostenveld’s template, picking out the appropriate channels to create a 3D channel location phantom for the data. Using this channel phantom the Montreal Brain Phantom could be warped to match the channels, giving us head tesselation surfaces for the forward model calculations. This head model is a very crude estimation of a subjects head but should be enough for the the analysis intended and introduced in the Method section above. Figure 6.1 shows the Montreal Brain Phantom warped to align with the phantom channels.

Section 3.4 gave an introduction on BEM head models and in section 3.5 the BrainStorm software package used to calculate the head models was introduced. For the sake of reproducibility all details regarding the BEM head model calculations here are listed in appendix C.2. The lead field matrix $\mathbf{A}$ for the BEM head model has dimensions $N = 118$ and $P = 10001$. Its condition number is $\kappa = 2.35 \times 10^8$, i.e. very ill-conditioned.

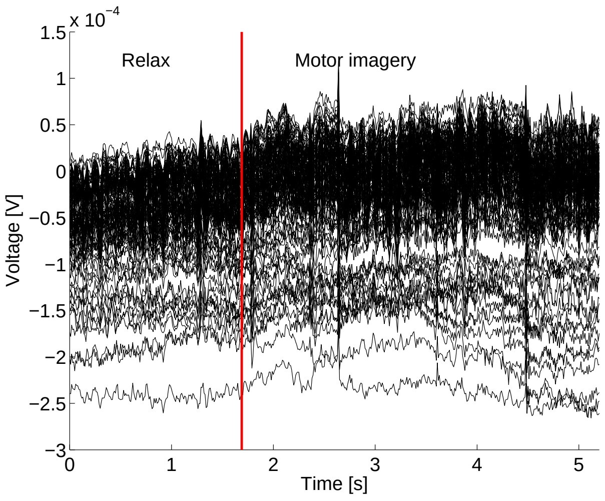

Figure 6.2: *Voltages of all 118 channels plotted on the same graph. The first 1.7s are a relax period and the last 3.5s are the motor imagery period. Solid vertical line separates the two periods on the graph.*

6.4 Results

Subject 2 from the dataset, marked *al*, is used. By visually inspecting the data a time window is selected where no obvious increase in noise or abnormally high voltage causing saturation in the measurements is present. Visual cues are indicated for the subject to perform motor imageries for a period of 3.5s. For the selected cue the subject should be visualizing right hand movement. Preceding the cue there is a relax period of at least 1.75s, the selected time window for this analysis is therefore 1.7s before the cue following the 3.5s period of motor imagery, giving a total time window of 5.2s. Figure 6.2 shows the EEG voltages plotted on a single graph, relax and motor imagery periods are separated by a solid vertical line.

The first goal is to show that there is a decrease in the mu rhythm, which is located in the alpha band, during motor imagery. We therefore look at the *discrete Fourier transforms (DFT's)* of the EEG channels during relaxation and during motor imagery. The Fourier transform gives information on the energy of different frequency components in the signal, using DFT we can therefore compare the energy difference before and during motor imagery. In appendix A.7 the DFT is explained further. Figure 6.3 shows the DFT squared amplitudes

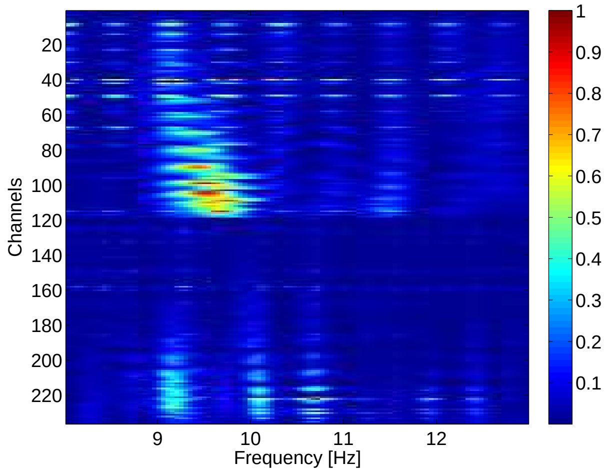

Figure 6.3: *Normalized squared DFT amplitudes of all channels during relaxation and motor imagery. The top half of the graph is during relaxation and the bottom half during motor imagery, i.e. if $k = 1, ..., 118$ then channels k and $k + 118$ are the same channel namely before (k) and during ($k + 118$) motor imagery. Frequency range shown is the alpha band (8-13Hz), where the mu rhythm is located.*

of all channels and gives a measure of the energy of different frequencies. The top half of the figure is during relaxation and the bottom half is during motor imagery. There is a clear reduction in energy during motor imagery indicating a reduction in the mu rhythm. The next step is therefore to find at which time the mu rhythm minimum is located. For that we use the *spectrograms* of the channels. Spectrogram is the short-time Fourier transform of a signal and can therefore be used as an estimate of the signal power over frequency and time. More details on spectrograms is provided in appendix A.7. On figure 6.4 the spectrograms of two channels are shown over the alpha band. One of them

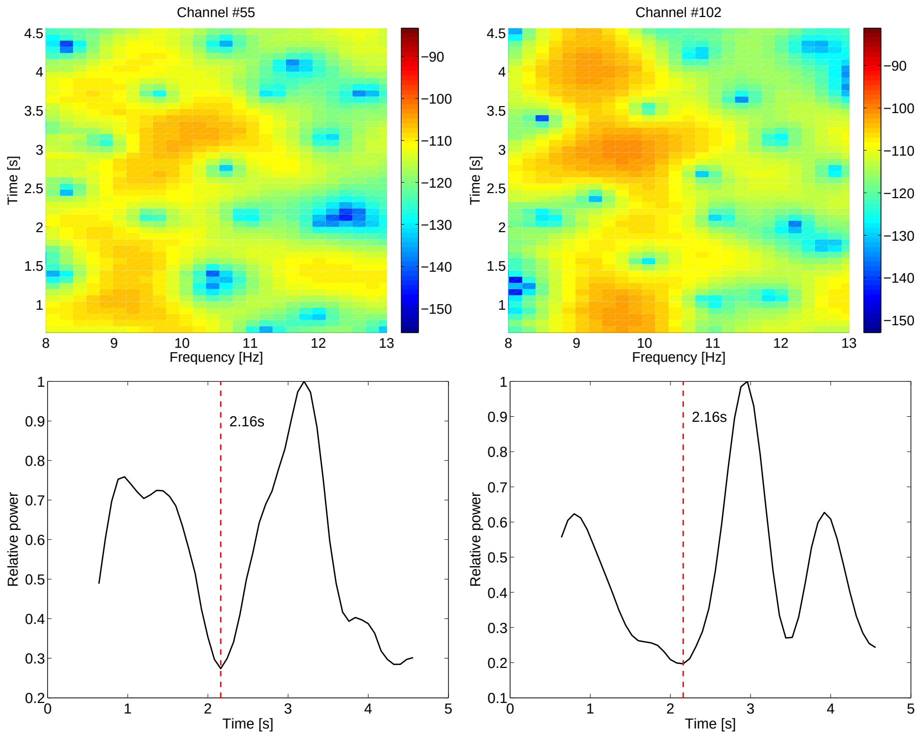

Figure 6.4: *Two top figures show spectrograms of channels #55 and #102, note that the amplitude scale is logarithmic. The minimum power in time is found by integrating over frequencies, showing a minimum power at 2.16s.*

(channel #55) is located over the motor cortex and the other (channel #102) is placed on the back of the head. Those channels were chosen because of the location and relative high mu activity respectively. We locate the minimum power in time by integrating over the frequencies which gives a measure of the alpha band power over time. This is shown for both channels on the two bottom graphs of figure 6.4. The minimum power turns out to be located at 2.16s. A rather wide Hamming window was used to calculate the spectrogram as the reduced time length of the power signals on figure 6.4 indicates. The width of the window determines how much is cut of from the start (0s) and end points (5.2s) of the time dimension.

Since the motor cortex activity is assumed to be linked with the mu rhythm located in the alpha band the data is filtered. A band pass filter with pass band edges at 8Hz and 13Hz is used to filter out all but the alpha rhythm of the EEG signals. Figure 6.5 shows the frequency response of the filter used. In the pass

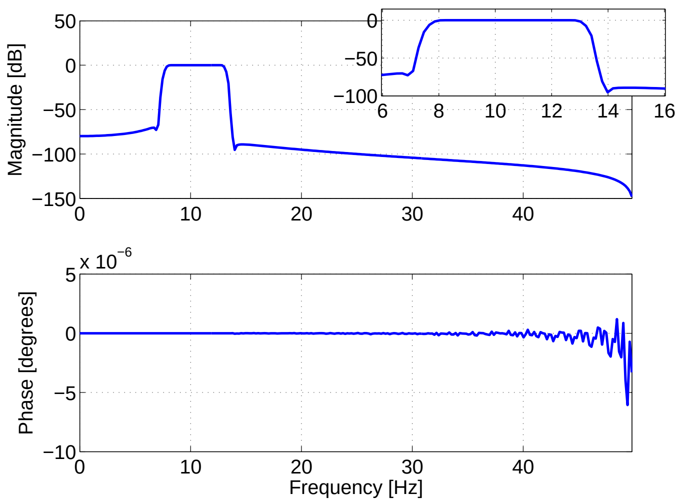

Figure 6.5: *Frequency response of the filter used to extract the alpha band from the EEG signals.*

band the gain is 0dB and the phase is 0 thus avoiding phase shift in the filtered signal. The filtered EEG signals are then used for the inverse calculations.

Now some quite extensive pre-processing of the data is finished and inverse calculations can be done using Algorithm I. The time period from 2.08s to 2.24s is chosen, i.e. around the 2.16s minimum value found using the spectrograms. Maximum number of iterations for the algorithm are set to 150 and the stop threshold at 10^{-3} making the number of iterations at each time instance vary only between 10 and 30 iterations. α and β converge nicely at all time instances and never go toward infinity. Figure 6.6 shows source estimates from Algorithm I for two different time slices, both brains on the figure have the right side turning down. On both estimates the activity is stronger on the right side of the brain and the one on the left has a clear maximum close to the center. Other activity on the front and back sides of the brain may be related to visual processing in the visual cortex and frontal lobe. Taking brain activity at a single time slice like this is not conclusive. On figure 6.7 the average activity over the time period from 2.08s to 2.24s is shown. Furthermore only activities larger than 80% of the largest activity strength are plotted, i.e. only the strongest activity is shown over the time period. This figure shows strong activity on the right side of the brain in the vicinity of the motor cortex. A strong pulse was also detected on the left side of the frontal lobe far away from the motor cortex. It was thus discarded in our analysis and is not shown on the figure. Some activity on the back side of the brain is also visible on the figure which may be related to visual processing. But our focus is on the activity near the motor cortex.

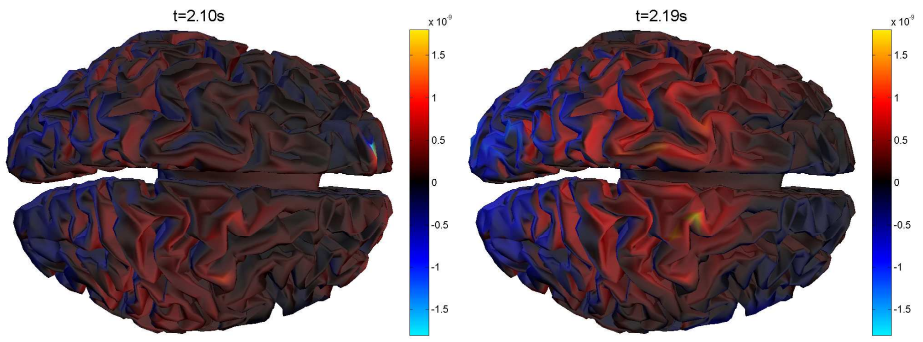

Figure 6.6: *Algorithm I estimates at times 2.10s and 2.19s. Right sides are more active at both times.*

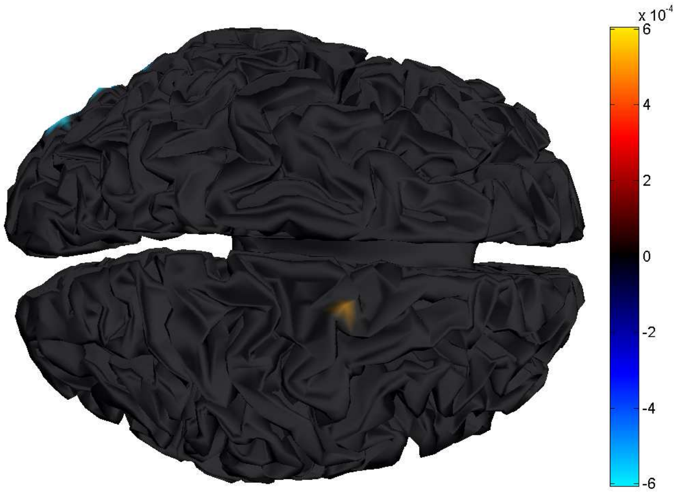

Figure 6.7: *Averaged Algorithm I estimates over the period from 2.08s to 2.24s with only the strongest activities shown. Strong activity is spotted on the right motor cortex (the yellow pulse). Left frontal lobe pulse was discarded and not plotted and some activity on the left half (blue pulse) of the brain can be seen, these extra activities may be related to visual processing.*

6.5 Summary

Using frequency analysis methods a decrease in the mu rhythm was located in time. This decrease was believed to be related to the subjects motor imagery on the right side of his body. A time window was chosen around this minimum. Using Algorithm I motor cortex activity was detected on the right side of the brain. This finding complies with the literature since motor imageries decrease the mu rhythm on the side which controls the movement. In our case this means that the mu rhythm is decreased on the left side and then stronger cortical activity is detected on the right side. Algorithm IId was also tested on the data but it did not converge very good at all time indices. At some time indices where convergence was good frontal and right temporal lobe activities were mainly detected.

Further work with this could be to try to detect a difference between the right hand and right foot motor imageries, e.g. with respect to the location of the strongest activity on the cortex. If that can be done in a robust way a classifier could be constructed and tested on the whole dataset.

Chapter 7

Conclusion

A Bayesian approach was used to solve the ill-posed EEG inverse problem. The likelihood distribution describing the noise was assumed Gaussian in all cases. Three different prior distributions were used. First a simple Gaussian prior distribution was assumed. Iterative update equations were derived and used to formulate Algorithm I. This algorithm estimates the sources and two hyperparameters. The hyperparameters describe the variances of the likelihood and prior. The update equations were shown to return non-negative hyperparameters. SVD and some simple linear algebra was used to numerically improve Algorithm I. This formulation is equivalent to a Tikhonov regularizer where the regularization parameter would be chosen as the ratio of the hyperparameters.

Automatic relevance determination (ARD) was used to form a prior distribution which assumes a more sparse sources space. With the ARD prior each source was assigned a hyperparameter, i.e. the sources are assumed independent of each other. The hyperparameters therefore indicate the relevance of their sources. The number of hyperparameters equals the number of sources plus the likelihood hyperparameter. The same approach was taken as for Algorithm I to derive update equations. These equations formulated Algorithm II. The update equation for the prior hyperparameters was quite compact and using some detailed algebra the numerical complexity of it was reduced extensively, this included deriving an equation to find only the necessary diagonal elements of the inverse Hessian matrix. The update equation was shown to return positive values. The author has not found this update equation elsewhere in the literature. Algorithm II also involved an update equation for the likelihood hyperparameter. This equation was shown to return non-negative values but its estimates were not as good as the ones from Algorithm I. That was the main reason for formulating Algorithm IIb. Algorithm IIb was constructed in the same way as Algorithm II except the likelihood hyperparameter was estimated using Algorithm I, improving the noise estimate. Algorithm IIc included further improvements by reducing the number of prior hyperparameters and their corresponding sources during iteration. This could be done since hyperparameters representing irrelevant sources approach infinity. The final variant of Algorithm II, called Algorithm IId, included a low-pass filtering on the sources. This low-pass filtering was based on the assumption that neighbouring sources are more likely to be active than ones placed further away from each other. The main reason for adding the low-pass filter was poor performance of Algorithm II, IIb and IIc when using realistic forward models.

Finally an attempt was made to incorporate spatial smoothing into the prior distribution. This involved some detailed calculations leading to overly complex equations for the hyperparameters. Formulating update equations for them where non-negativity would be guaranteed was not attempted. Instead this prior assumption was only used to estimate the sources and the hyperparameters were estimated in the same way as in Algorithm IIb. This mixture of ARD and a smoothing prior formulated Algorithm III. Algorithm III had similar performance as Algorithm IId but did not outperform it.

Simulations were done to evaluate the algorithms. For sources with distribution close to being Gaussian Algorithm I could locate a localized increased activity on a Gaussian floor. For well-posed systems Algorithm I estimated source and noise variances perfectly. But for extremely low variances the estimates converged to a constant value, most likely due to the finite numerical resolution in computer calculations. For ill-posed systems the noise estimate got worse, usually underestimating the noise. The source distribution variance was accurately predicted for ill-posed systems. The condition number of the lead field matrix did not have considerable effect on the algorithm. Algorithm II noise estimate was generally worse than the noise estimate from Algorithm I, but for reasonably good SNR and well-conditioned lead field matrix its source estimate was good. With improved noise estimate in Algorithm IIb the performance got better. By removing irrelevant sources during iteration in Algorithm IIc the calculation time decreased but it was shown that one would have to be careful not to choose too low threshold value which controlled the removal boundary. For ill-conditioned realistic lead field matrices Algorithms II, IIb and IIc gave very poor results. Algorithm IId did however give useful results by locating sources but loosing a bit much of their volume. Algorithm III worked very much like Algorithm IId but for reasonable SNR values Algorithm IId performed better.

SNR was shown to have a great effect. For well-conditioned lead-field matrices Algorithm IId gave the best results and below 0dB all algorithm estimates soon became useless. For realistic ill-conditioned lead field matrices Algorithm I however performed best in general and only Algorithms IId and III came close to its performance. We can therefore confidently conclude that the use of Algorithm I must be preferred over the more complex ones when analyzing real EEG recordings. It is robust, converges much faster than the others and gives good estimates even when the source distribution is far from being Gaussian.

Real EEG scalp recordings were analyzed. Dataset IVa from the BCI Competition III was used and a selected time period around a single motor imagery was chosen. Detailed pre-processing revealed a time index where right motor cortex activity was expected. This activity was confirmed by solving the inverse problem around the selected time using Algorithm I. Inverse calculations were also done using Algorithm IId but its estimate was too sparse, only showing frontal and temporal lobe activities which were also present in the estimate from Algorithm I. Some real data testing was also presented in appendix D where Algorithm I and IId both converged nicely. Their source estimates were very similar with the Algorithm IId estimate being more sparse.

Overall we can summarize that different algorithms were derived using Bayes theorem. The simplest one, Algorithm I, based on a Gaussian distribution of the source space had the best performance for realistic forward models. The more complex algorithms, based on ARD, gave far better results than Algorithm I for well-conditioned artificial forward models. To justify their use on ill-conditioned realistic forward models some additional constraints or simplifications are needed to improve their source estimates.

7.1 Future Work

The inverse EEG problem is vast and can be divided into many different parts. An introduction to the forward modeling was presented in the thesis. That part of the EEG problem is a research field on its own. Forward model improvements naturally improve inverse methods. The Bayesian approach to the inverse problem has great potential and there are many ways the framework presented here can be extended or tackled differently. Here a short list of ideas is therefore presented.

The first thing one thinks of is including a time index into the forward model, as shown in equation 3.12. This would allow temporal smoothing to be added to the model and noise estimation for each channel. Some computational overhead would be expected.

Well suited EEG recordings for the initial tests of the Algorithms were hard to come by. Testing the ARD algorithms on averaged evoked potential (EP) data would be of interest since background brain activity should be averaged out.

Adding some kind of smoothing into Algorithm I should give an algorithm very similar to the LORETA method, with the advantage of automatically determining the regularization parameter.

Anatomical information could be included by labeling the different areas of the cortex. This could provide additional grouping of sources which could be added to the prior distribution. Non-anatomical grouping, possibly overlapping, could also be included into the prior to produce more localized estimates.

Algorithm I could be used to locate the most active areas of the cortex. That way a subset of the source space could be labeled and Algorithm IId run on the subspace, giving a more localized solution. A too simplified approach to this was attempted when analyzing the BCI data. Only a single one of the most active areas was labeled and strong pulses were left out. Algorithm IId did not give good estimates when run on this subset.

Constricting the source grid to the cortical surface is a good approximation since most of the activity measured by EEG is from the surface. Some activity is however in deeper areas in the brain and in studies focusing on them it is necessary to expand the source space to 3D coordinates inside the cortical tesselation surface.

Appendix A

Mathematical Appendix

Here some mathematical details are presented. Some sections give additional information on topics which were mentioned in the main text without giving a detailed discussion. Other sections provide the details of some tedious calculations which are used in some of the derivations in the thesis. The first section lists the main mathematical symbols used throughout the text.

A.1 Nomenclature

Here the most common symbols from the text are listed.

$\mathbf{r}$	Point in space
$\mathbf{r}_q$	Dipole location
$\mathbf{q}$	Current dipole
$\mathbf{J}^p(\mathbf{r})$	Impressed/primary current
$\mathbf{J}^v(\mathbf{r})$	Volume current
σ	Conductivity
$\mathbf{E}(\mathbf{r})$	Electric field
$\delta(\mathbf{r})$	Dirac delta function
G_i	Volume i
S_i	Surface bounding volume G_i
$V(\mathbf{r})$	Potential at a certain point
$V^1(\mathbf{r})$	Single sphere surface potential
$V^M(\mathbf{r})$	Multishell surface potential
$m(\mathbf{r})$	Scalp potential
$\mathbf{m}$	Vector of scalp potentials
N	Dimension of $\mathbf{m}$
$\mathbf{s}$	Vector of dipole magnitudes
P	Dimension of $\mathbf{s}$
$\mathbf{A}$	Lead-field matrix (gain matrix)
$N \times P$	Dimensions of $\mathbf{A}$
$\boldsymbol{\epsilon}$	Noise vector
$p(\cdot)$	Probability distribution
α	Prior distribution hyperparameter
β	Likelihood distribution hyperparameter
E_s	Decay function
E_m	Error function
σ_ϵ^2	Noise variance
$\sigma_{p(m\|s)}^2$	Likelihood variance
$\sigma_{p(s)}^2$	Prior variance
$\mathbf{H}$	Hessian matrix
h'_{kk}	k-th diagonal element of the inverse Hessian
$\mathbf{I}$	Identity matrix
$\boldsymbol{\Lambda}$	Diagonal matrix of hyperparameters
$\boldsymbol{\Gamma}$	Diagonal matrix of hyperparameters

A.2 MEG Forward Model

Magnetoencephalography (MEG) is related measurement technique to EEG. It measures the magnetic field, $\mathbf{B}(\mathbf{r})$, outside the head volume using multichannel SQUID (superconducting quantum interference device) gradiometers. Same methods as used for EEG are used when solving the MEG inverse problem. EEG and MEG are complementary and often used together. For detailed review on MEG see [15].

Here we start in the same way as in section 3.2, i.e we know that the total current density at each point in the head volume can be divided into two components

$$\mathbf{J}(\mathbf{r}) = \mathbf{J}^p(\mathbf{r}) + \mathbf{J}^v(\mathbf{r}),$$

where $\mathbf{J}^p(\mathbf{r})$ is the primary current and $\mathbf{J}^v(\mathbf{r})$ is the passive current. Studies deal with frequencies below 100 Hz allowing the physics to be described by using the quasi-static approximation of Maxwell's equations [21]. That means that the electric field can be expressed with a scalar potential

$$\mathbf{E}(\mathbf{r}) = -\nabla V(\mathbf{r})$$

and then the current can be written as

$$\mathbf{J}(\mathbf{r}) = \mathbf{J}^p(\mathbf{r}) - \sigma(\mathbf{r})\nabla V(\mathbf{r}),$$

Now the current flow $\mathbf{J}(\mathbf{r}_q)$ at location $\mathbf{r}_q$ can be related to the magnetic field $\mathbf{B}(\mathbf{r})$ at location $\mathbf{r}$ through the Biot-Savart law

$$\mathbf{B}(\mathbf{r}) = \frac{\mu_0}{4\pi}\int_G \mathbf{J}(\mathbf{r}_q) \times \frac{\mathbf{r}-\mathbf{r}_q}{|\mathbf{r}-\mathbf{r}_q|^3} dv. \tag{A.1}$$

This equation can be rewritten using the identity

$$\frac{\mathbf{r}-\mathbf{r}_q}{|\mathbf{r}-\mathbf{r}_q|^3} = -\nabla\left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|}\right)$$

giving

$$\mathbf{B}(\mathbf{r}) = -\frac{\mu_0}{4\pi}\int_G \mathbf{J}(\mathbf{r}_q) \times \nabla\left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|}\right) dv. \tag{A.2}$$

Now lets use equation 3.3 for the current density, giving

$$\begin{aligned}\mathbf{B}(\mathbf{r}) &= -\frac{\mu_0}{4\pi}\int_G (\mathbf{J}^p(\mathbf{r}_q) - \sigma(\mathbf{r}_q)\nabla V(\mathbf{r}_q)) \times \nabla\left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|}\right) dv \\ &= \frac{\mu_0}{4\pi}\int_G \mathbf{J}^p(\mathbf{r}_q) \times \frac{\mathbf{r}-\mathbf{r}_q}{|\mathbf{r}-\mathbf{r}_q|^3} dv + \frac{\mu_0}{4\pi}\int_G \sigma(\mathbf{r}_q)\nabla V(\mathbf{r}_q) \times \nabla\left(\frac{1}{|\mathbf{r}-\mathbf{r}_q|}\right) dv.\end{aligned}$$

If we assume the head consists of different regions (brain,skull,scalp, etc.) of uniform and isotropic conductances, σ_i, we can write

$$\mathbf{B}(\mathbf{r}) = \mathbf{B_0}(\mathbf{r}) + \frac{\mu_0}{4\pi} \sum_i \int_{G_i} \sigma_i(\mathbf{r}_q) \nabla V(\mathbf{r}_q) \times \nabla \left(\frac{1}{|\mathbf{r} - \mathbf{r}_q|} \right) dv \tag{A.3}$$

where primary magnetic field $\mathbf{B_0}(\mathbf{r})$ is only related to the primary current $\mathbf{J}^p(\mathbf{r}_q)$

$$\mathbf{B_0}(\mathbf{r}) = \frac{\mu_0}{4\pi} \int_G \mathbf{J}^p(\mathbf{r}_q) \times \frac{\mathbf{r} - \mathbf{r}_q}{|\mathbf{r} - \mathbf{r}_q|^3} dv. \tag{A.4}$$

This equation can be transformed from a volume integral to a surface integral, but first lets look at an identity needed to do so. We can write

$$\begin{aligned} \nabla \times \nabla \left(\frac{V}{|\mathbf{r} - \mathbf{r}_q|} \right) = 0 \quad &= \quad \nabla \times \left(\nabla \frac{V}{|\mathbf{r} - \mathbf{r}_q|} + V \nabla \frac{1}{|\mathbf{r} - \mathbf{r}_q|} \right) \\ &= \quad -\nabla V \times \nabla \frac{1}{|\mathbf{r} - \mathbf{r}_q|} + \nabla \times \left(V \nabla \frac{1}{|\mathbf{r} - \mathbf{r}_q|} \right) \end{aligned}$$

which gives

$$\nabla V \times \nabla \frac{1}{|\mathbf{r} - \mathbf{r}_q|} = \nabla \times \left(V \nabla \frac{1}{|\mathbf{r} - \mathbf{r}_q|} \right).$$

Using this identity along with Stoke's theorem on the integral in equation A.3 gives

$$\begin{aligned} \int_{G_i} \sigma_i(\mathbf{r}_q) \nabla V \times \nabla \left(\frac{1}{|\mathbf{r} - \mathbf{r}_q|} \right) dv \quad &= \quad \int_{G_i} \nabla \times V \nabla \left(\frac{1}{|\mathbf{r} - \mathbf{r}_q|} \right) dv \\ &= \quad -\int_{S_i} V \nabla \left(\frac{1}{|\mathbf{r} - \mathbf{r}_q|} \right) \times d\mathbf{s} \end{aligned}$$

where S_i is the surface bounding volume G_i. Now using boundary conditions of continuous voltages and currents on an interface between regions of conductivities σ' and σ'', i.e.

$$V' = V'' \quad \text{and} \quad \sigma' \delta V' / \delta n = \sigma'' \delta V'' / \delta n \quad \text{on } S_i,$$

we get

$$\mathbf{B}(\mathbf{r}) = \mathbf{B_0}(\mathbf{r}) - \frac{\mu_0}{4\pi} \sum_i \int_{S_i} (\sigma' - \sigma'') V(\mathbf{r}_q) \nabla \left(\frac{1}{|\mathbf{r} - \mathbf{r}_q|} \right) \times d\mathbf{s}_i \tag{A.5}$$

where σ' and σ'' are the conductivities of the inner and outer sides of S_i respectively, as illustrated on figure 3.2. From here on we assume that $d\mathbf{s}_i$ is directed from the primed region to the double primed. And at the external boundary $\sigma'' = 0$ (i.e. the air conductivity). The equation above states that if we know the primary current distribution, $\mathbf{J}^p$, and the potential, V, on all surfaces we can calculate the magnetic field, $\mathbf{B}(\mathbf{r})$, at location $\mathbf{r}$. In the case of MEG measurements then $\mathbf{r}$ is placed outside of the head surface. This also shows that to solve the MEG forward model the potentials on all surfaces have to be known, i.e. one also has to solve the EEG forward model presented in section 3.2.

A.3 Derivative and Hessian of $L(\mathbf{s})$

A.3.1 Two Hyperparameters Case, $L(\mathbf{s}, \alpha, \beta)$

Here the calculations leading to equation 4.18 will be given in more detail. They are straight forward but tedious.

The goal is to calculate the derivative

$$\frac{\partial}{\partial \mathbf{s}} L(\mathbf{s}) = \frac{\partial}{\partial \mathbf{s}} \left(\frac{\beta}{2} |\mathbf{m} - \mathbf{A}\mathbf{s}|^2 \right) + \frac{\partial}{\partial \mathbf{s}} \left(\frac{\alpha}{2} |\mathbf{s}|^2 \right).$$

Lets first take a look at the second term on the right side. We get

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{s}} \left(\frac{\alpha}{2} |\mathbf{s}|^2 \right) &= \frac{\alpha}{2} \frac{\partial}{\partial \mathbf{s}} \sum_{i=1}^{P} s_i^2 \\
&= \frac{\alpha}{2} \begin{bmatrix} \frac{\partial}{\partial s_1} \sum_{i=1}^{P} s_i^2 \\ \vdots \\ \frac{\partial}{\partial s_P} \sum_{i=1}^{P} s_i^2 \end{bmatrix} \\
&= \frac{\alpha}{2} \begin{bmatrix} 2s_1 \\ \vdots \\ 2s_P^2 \end{bmatrix} \\
&= \alpha \mathbf{s}.
\end{aligned}$$

And then the more involved part of calculating the first derivative on the right. Using the notation

$$\mathbf{A} = \begin{bmatrix} a_{11} & \dots & a_{1P} \\ \vdots & \ddots & \vdots \\ s_{N1} & \dots & a_{NP} \end{bmatrix}, \quad \mathbf{s} = \begin{bmatrix} s_1 \\ \vdots \\ s_P \end{bmatrix}, \quad \mathbf{m} = \begin{bmatrix} m_1 \\ \vdots \\ m_N \end{bmatrix}$$

and some basic algebra gives

$$
\begin{aligned}
& \tfrac{\partial}{\partial \mathbf{s}}\left(\tfrac{\beta}{2}|\mathbf{m}-\mathbf{A}\mathbf{s}|^2\right) \\
= \; & \tfrac{\beta}{2}\tfrac{\partial}{\partial \mathbf{s}}\left(\textstyle\sum_{n=1}^{N}(m_n - a_{n1}s_1 - a_{n2}s_2 - ... - a_{nP}s_P)^2\right) \\
= \; & \tfrac{\beta}{2}\begin{bmatrix} \tfrac{\partial}{\partial s_1}\left(\sum_{n=1}^{N}(m_n - a_{n1}s_1 - a_{n2}s_2 - ... - a_{np}s_P)^2\right) \\ \tfrac{\partial}{\partial s_2}\left(\sum_{n=1}^{N}(m_n - a_{n1}s_1 - a_{n2}s_2 - ... - a_{np}s_P)^2\right) \\ \vdots \\ \tfrac{\partial}{\partial s_P}\left(\sum_{n=1}^{N}(m_n - a_{n1}s_1 - a_{n2}s_2 - ... - a_{np}s_P)^2\right) \end{bmatrix} \\
= \; & -\beta\begin{bmatrix} \sum_{n=1}^{N}(m_n - a_{n1}s_1 - a_{n2}s_2 - ... - a_{np}s_P)a_{n1} \\ \sum_{n=1}^{N}(m_n - a_{n1}s_1 - a_{n2}s_2 - ... - a_{np}s_P)a_{n2} \\ \vdots \\ \sum_{n=1}^{N}(m_n - a_{n1}s_1 - a_{n2}s_2 - ... - a_{np}s_P)a_{nP} \end{bmatrix} \\
= \; & -\beta\begin{bmatrix} \sum_{n=1}^{N} m_n a_{n1} - \sum_{n=1}^{N} a_{n1}(a_{n1}s_1 + ... + a_{nP}s_P) \\ \sum_{n=1}^{N} m_n a_{n2} - \sum_{n=1}^{N} a_{n2}(a_{n1}s_1 + ... + a_{nP}s_P) \\ \vdots \\ \sum_{n=1}^{N} m_n a_{np} - \sum_{n=1}^{N} a_{nP}(a_{n1}s_1 + ... + a_{nP}s_P) \end{bmatrix} \\
= \; & -\beta\begin{bmatrix} m_1a_{11} + m_2a_{21} + ...m_Na_{N1} \\ m_1a_{12} + m_2a_{22} + ...m_Na_{N2} \\ \vdots \\ m_1a_{1P} + m_2a_{2P} + ...m_Na_{NP} \end{bmatrix} + \\
& \beta\begin{bmatrix} a_{11}(a_{11}s_1 + ... + a_{1P}s_P) + ... + a_{N1}(a_{N1}s_1 + ... + a_{NP}s_P) \\ a_{12}(a_{11}s_1 + ... + a_{1P}s_P) + ... + a_{N2}(a_{N1}s_1 + ... + a_{NP}s_P) \\ \vdots \\ a_{1P}(a_{11}s_1 + ... + a_{1P}s_P) + ... + a_{NP}(a_{N1}s_1 + ... + a_{NP}s_P) \end{bmatrix} \\
= \; & -\beta\left(\mathbf{m}^T\mathbf{A}\right)^T + \beta\begin{bmatrix} a_{11} & \cdots & a_{N1} \\ \vdots & \ddots & \vdots \\ a_{1P} & \cdots & a_{NP} \end{bmatrix}\begin{bmatrix} a_{11}s_1 + a_{12}s_2 + ... + a_{1P}s_P \\ \vdots \\ a_{N1}s_1 + a_{N2}s_2 + ... + a_{NP}s_P \end{bmatrix} \\
= \; & -\beta(\mathbf{A}^T\mathbf{m} - \mathbf{A}^T\mathbf{A}\mathbf{s}).
\end{aligned}
$$

Now summing the two derivatives gives

$$\frac{\partial}{\partial \mathbf{s}}L(\mathbf{s}) = -\beta(\mathbf{A}^T\mathbf{m} - \mathbf{A}^T\mathbf{A}\mathbf{s}) + \alpha\mathbf{s}. \tag{A.6}$$

We are also interested in the Hessian matrix of $L(\mathbf{s})$. Lets take a look at the value in line $l = 1, .., P$ and row $k = 1, .., P$ of the Hessian and call it h_{lk}, i.e.

$$h_{lk} = \frac{\partial^2 L(\mathbf{s})}{\partial s_l \partial s_k} = \beta\frac{\partial^2 E_m}{\partial s_l \partial s_k} + \alpha\frac{\partial^2 E_s}{\partial s_l \partial s_k}$$

where

$$\frac{\partial^2}{\partial s_l \partial s_k} E_m = a_{1l}a_{1k} + a_{2l}a_{2k} + ... + a_{Nl}a_{Nk}$$

and

$$\frac{\partial^2}{\partial s_l \partial s_k} E_s = \begin{cases} 1 & \text{if } l = k \\ 0 & \text{if } l \neq k \end{cases}$$

Rewriting this to matrices gives

$$\mathbf{H} = \beta \mathbf{A}^T \mathbf{A} + \alpha \mathbf{I}. \tag{A.7}$$

A.3.2 ARD Case, $L(\mathbf{s}, \mathbf{\Lambda}, \beta)$

Here we want to find the derivative of

$$L(\mathbf{s}, \alpha, \beta) = \frac{\beta}{2}|\mathbf{m} - \mathbf{A}\mathbf{s}|^2 + \frac{1}{2}\mathbf{s}^T \mathbf{\Lambda} \mathbf{s}.$$

where $L(\mathbf{s}, \alpha, \beta)$ is given in equation 4.47. From above we have

$$\frac{\partial}{\partial \mathbf{s}} \left(\frac{\beta}{2} |\mathbf{m} - \mathbf{A}\mathbf{s}|^2 \right) = -\beta(\mathbf{A}^T \mathbf{m} - \mathbf{A}^T \mathbf{A}\mathbf{s}).$$

Then lets look at the other derivate needed

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{s}} \left(\frac{1}{2} \mathbf{s}^T \mathbf{\Lambda} \mathbf{s} \right) &= \frac{1}{2} \frac{\partial}{\partial \mathbf{s}} \sum_{i=1}^{P} \alpha_i s_i^2 \\
&= \frac{1}{2} \begin{bmatrix} \frac{\partial}{\partial s_1} \sum_{i=1}^{P} \alpha_i s_i^2 \\ \vdots \\ \frac{\partial}{\partial s_P} \sum_{i=1}^{P} \alpha_i s_i^2 \end{bmatrix} \\
&= \frac{1}{2} \begin{bmatrix} 2\alpha_1 s_1 \\ \vdots \\ 2\alpha_P s_P^2 \end{bmatrix} \\
&= \begin{bmatrix} \alpha_1 s_1 \\ \vdots \\ \alpha_P s_P^2 \end{bmatrix} \\
&= \mathbf{\Lambda}\mathbf{s}.
\end{aligned}$$

So now we have

$$\frac{\partial}{\partial \mathbf{s}} L(\mathbf{s}) = -\beta(\mathbf{A}^T \mathbf{m} - \mathbf{A}^T \mathbf{A}\mathbf{s}) + \mathbf{\Lambda}\mathbf{s}. \tag{A.8}$$

A.4 Derivative of $\ln\det(\mathbf{H})$

In section 4.2.1 following derivative is needed

$$\frac{\partial}{\partial \alpha_k} \ln\det(\mathbf{H}).$$

where the Hessian matrix $\mathbf{H}$ is

$$\mathbf{H} = \beta \mathbf{A}^T \mathbf{A} + \mathbf{\Lambda}$$

Its calculations are a bit tedious and therefore listed here in appendix for the interested reader. We can write

$$\frac{\partial}{\partial \alpha_k} \ln\det(\mathbf{H}) = \mathrm{Tr}\left(\mathbf{H}^{-1} \frac{\partial}{\partial \alpha_k} \mathbf{H}\right).$$

Lets first look at

$$\begin{aligned} \frac{\partial}{\partial \alpha_k}\mathbf{H} &= \frac{\partial}{\partial \alpha_k}\left(\beta \mathbf{A}^T \mathbf{A} + \mathbf{\Lambda}\right) \\ &= \frac{\partial}{\partial \alpha_k}\mathbf{\Lambda} \end{aligned} \tag{A.9}$$

which is a matrix with elements d_{ij} where

$$d_{ij} = \begin{cases} 1 & \text{if } i = j = k \\ 0 & \text{otherwise} \end{cases}$$

If we now write the inverse of the Hessian as

$$\mathbf{H}^{-1} = \begin{bmatrix} h'_{11} & h'_{12} & \cdots & h'_{1P} \\ h'_{21} & h'_{22} & \cdots & h'_{2P} \\ \vdots & \vdots & \ddots & \vdots \\ h'_{P1} & h'_{P2} & \cdots & h'_{PP} \end{bmatrix}$$

we get

$$\mathbf{H}^{-1}\frac{\partial}{\partial \alpha_k}\mathbf{H} = \begin{bmatrix} & h'_{1k} & \\ \mathbf{0} & \vdots & \mathbf{0} \\ & h'_{Pk} & \end{bmatrix}$$

which is a $P \times P$ matrix with zeros elements everywhere except in the k-th column. Then finally we can write

$$\begin{aligned} \frac{\partial}{\partial \alpha_k} \ln\det(\mathbf{H}) &= \mathrm{Tr}\left(\begin{bmatrix} & h'_{1k} & \\ \mathbf{0} & \vdots & \mathbf{0} \\ & h'_{Pk} & \end{bmatrix}\right) \\ &= h'_{kk}. \end{aligned} \tag{A.10}$$

A.5 Algorithm II - Multiplicity of α_{k-new}

The update equation for hyperparameters α_{k-new} in Algorithm II is

$$\alpha_{k,new} = \frac{1}{s_k^2 + h'_{kk}}.$$

In the main text it is shown that $\alpha_{k,new} > 0$. The equation was formulated from the derivative of the log-likelihood given by equation 4.55, rewritten here for clarity

$$\frac{\partial}{\partial \alpha_k} \ln p(\mathbf{m}|\mathbf{\Lambda}, \beta) = \frac{1}{2\alpha_k} - \frac{s_k^2}{2} - \frac{h'_{kk}}{2}.$$

Now lets formulate the update equation in a multiplicative way and show that the result is the same as before. We write

$$\alpha_{k,new} = \alpha_k + \xi \frac{\partial E}{\partial \alpha_k}$$

where $E = \ln p(\mathbf{m}|\mathbf{\Lambda}, \beta)$. Then

$$\begin{aligned} \alpha_{k,new} &= \alpha_k + \xi \left(\frac{1}{2\alpha_k} - \frac{s_k^2}{2} - \frac{h'_{kk}}{2} \right) \\ &= \alpha_k - \xi \left(\frac{s_k^2 + h'_{kk}}{2} - \frac{1}{2\alpha_k} \right) \end{aligned}$$

put

$$\xi = \frac{\alpha_k}{(s_k^2 + h'_{kk})/2}$$

then

$$\begin{aligned} \alpha_{k,new} &= \alpha_k - \alpha_k + \frac{\alpha_k}{((s_k^2 + h'_{kk})/2)} \frac{1}{2\alpha_k} \\ &= \frac{1}{s_k^2 + h'_{kk}} \end{aligned} \tag{A.11}$$

which is the same as the update from Algorithm II.

A.6 Framework for Hyperparameters α_k and β

Here we continue with the work from section 4.3 and derive equations that form a framework for hyperparameters α_k $(k = 1, ..., P)$ and β in the case of a smoothing prior distribution. The equations derived here are similar to the ones from the ARD formulation (section 4.2.1) but include some tedious summations which are very impractical for computer implementation and more importantly formulating update equations for α_k $(k = 1, ..., P)$ where non-negativity is guaranteed is hard. These were the main reasons for not using them when formulating Algorithm III, they are however presented here for the interested reader and possible future use.

Gaussian expansion of $L(\mathbf{s})$ around the most probable value $\mathbf{s}_{MP}$ is used.

$$L(\mathbf{s}) = L(\mathbf{s}_{MP}) + \frac{1}{2}(\mathbf{s} - \mathbf{s}_{MP})^T \mathbf{H}(\mathbf{s} - \mathbf{s}_{MP}).$$

The Hessian matrix is

$$\begin{aligned} \mathbf{H} &= \nabla\nabla L(\mathbf{s}, \mathbf{\Lambda}, \beta) \\ &= \beta \mathbf{A}^T \mathbf{A} + \nabla\nabla \left(-\frac{1}{2} \sum_{i=1}^{P} (\alpha_i + \psi_i) s_i^2 \right) \\ &= \beta \mathbf{A}^T \mathbf{A} + \mathbf{\Gamma} \end{aligned} \tag{A.12}$$

and the posterior distribution and normalization constant can be written (equations 4.22 and 4.23)

$$\begin{aligned} p(\mathbf{s}|\mathbf{m}) &= \frac{1}{Z_L^*} \exp\left(-L(\mathbf{s}_{MP}) - \frac{1}{2}(\mathbf{s} - \mathbf{s}_{MP})^T \mathbf{H}(\mathbf{s} - \mathbf{s}_{MP}) \right) \\ Z_L^* &= \exp(-L(\mathbf{s}_{MP}))(2\pi)^{P/2} \det(\mathbf{H})^{-1/2}. \end{aligned}$$

Now the hyperparameters need to be estimated with the best generalization in mind as was done in section 4.1.1. To find the most probable values for α_k $(k = 1, ..., P)$ and β we evaluate

$$p(\alpha_1, ...\alpha_P, \beta|\mathbf{m}) = \frac{p(\mathbf{m}|\alpha_1, ...\alpha_P, \beta) p(\alpha_1, ...\alpha_P, \beta)}{p(\mathbf{m})} \tag{A.13}$$

assuming a flat non-informative hyperprior $p(\alpha_1, ...\alpha_P, \beta)$. That way maximizing $p(\alpha_1, ...\alpha_P, \beta|\mathbf{m})$ can be done by maximizing the evidence $p(\mathbf{m}|\alpha_1, ...\alpha_P, \beta)$.

Using $Z_L = Z_L^*$ along with previously derived identities gives gives

$$\begin{aligned} p(\mathbf{m}|\alpha_1, ...\alpha_P, \beta) &= \int p(\mathbf{m}|\mathbf{s}, \alpha_1, ...\alpha_P, \beta) p(\mathbf{s}|\alpha_1, ...\alpha_P, \beta) d\mathbf{s} \\ &= \int p(\mathbf{m}|\mathbf{s}, \beta) p(\mathbf{s}|\alpha_1, ...\alpha_P d\mathbf{s} \\ &= \left(\frac{2\pi}{\beta}\right)^{-N/2} \prod_{i=1}^{P} (\alpha_i + \psi_i)^{1/2} \exp(-L(\mathbf{s}_{MP})) \det(\mathbf{H})^{-1/2}. \end{aligned}$$

The natural logarithm of the evidence is then

$$\ln p(\mathbf{m}|\alpha_1, ...\alpha_P, \beta) = -\frac{N}{2} \ln \frac{2\pi}{\beta} + \frac{1}{2} \sum_{i=1}^{P} \ln(\alpha_i + \psi_i) - L(\mathbf{s}_{MP}) - \frac{1}{2} \ln \det(\mathbf{H}). \tag{A.14}$$

An identity for β which maximizes the log-evidence becomes exactly the same as derived in section 4.2.1 and given by equation 4.58.

Next an identity for α_k which maximizes the log-evidence is derived. Remember that each ψ_k is a function of a subset of the set containing all α_i except α_k, i.e. $\alpha_1, ..., \alpha_{k-1}, \alpha_{k+1}, ..., \alpha_P$ (or equivalently in set notation $\{\alpha_1, ..., \alpha_P\} \backslash \{\alpha_k\}$). We then find the derivatives of the three α_k dependent terms of the log-evidence. The first one is

$$\frac{\partial}{\partial \alpha_k} \left(\frac{1}{2} \sum_{i=1}^{P} \ln(\alpha_i + \psi_i) \right) = \frac{1}{2} \left(\frac{1}{\alpha_k + \psi_k} + \sum_{\substack{\text{all } \psi_j(\alpha_k) \\ j \neq k}} \frac{1}{n_k(\alpha_j + \psi_j(\alpha_k))} \right). \tag{A.15}$$

And the second one

$$\begin{aligned} \frac{\partial}{\partial \alpha_k} L(\mathbf{s}_{MP}) &= \frac{\partial}{\partial \alpha_k} \left(\frac{\beta}{2} |\mathbf{m} - \mathbf{A}\mathbf{s}|^2 + \frac{1}{2} \mathbf{s}^T \mathbf{\Gamma} \mathbf{s} \right) \\ &= \frac{\partial}{\partial \alpha_k} \left(\frac{1}{2} \sum_{i=1}^{P} (\alpha_i + \psi_i) s_i^2 \right) \\ &= \frac{1}{2} \left(s_k^2 + \sum_{\substack{\text{all } \psi_j(\alpha_k) \\ j \neq k}} \frac{\partial}{\partial \alpha_k} \psi_j(\alpha_k) s_j^2 \right) \\ &= \frac{1}{2} \left(s_k^2 + \sum_{\substack{\text{all } \psi_j(\alpha_k) \\ j \neq k}} \frac{s_j^2}{n_k} \right). \end{aligned} \tag{A.16}$$

Finally the third term

$$\frac{\partial}{\partial \alpha_k}\frac{1}{2}\ln\det(\mathbf{H}) = \frac{1}{2}\mathrm{Tr}\left(\mathbf{H}^{-1}\frac{\partial}{\partial \alpha_k}\mathbf{H}\right)$$

where

$$\begin{aligned}
\frac{\partial}{\partial \alpha_k}\mathbf{H} &= \frac{\partial}{\partial \alpha_k}\left(\beta \mathbf{A}^T\mathbf{A} + \mathbf{\Gamma}\right) = \frac{\partial}{\partial \alpha_k}\mathbf{\Gamma} \\
&= \frac{\partial}{\partial \alpha_k}\begin{bmatrix} \alpha_1 + \psi_i & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \alpha_P + \psi_P \end{bmatrix} \\
&\equiv \mathbf{D}
\end{aligned}$$

which is a diagonal matrix with elements d_{ii} where

$$d_{ii} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{if } i \neq k \text{ and } \psi \text{ depends on } \alpha_k \\ \frac{1}{n_k} & \text{if } i \neq k \text{ and } \psi \text{ not depends on } \alpha_k) \end{cases}$$

Let h'_{ij} be the elements of the inverse Hessian matrix $\mathbf{H}^{-1}$, then

$$\frac{1}{2}\mathrm{Tr}\left(\mathbf{H}^{-1}\frac{\partial}{\partial \alpha_k}\mathbf{H}\right) = \frac{1}{2}\left(h'_{kk} + \sum_{\substack{\text{all } \psi_j(\alpha_k) \\ j \neq k}} \frac{h'_{jj}}{n_j}\right).$$

Taking all three terms together then gives

$$\frac{\partial}{\partial \alpha_k}\ln p(\mathbf{m}|\alpha_1, ...\alpha_P, \beta) =$$

$$\frac{1}{2}\left(\frac{1}{\alpha_k + \psi_k} - s_k^2 - h'_{kk} + \sum\nolimits_{\substack{\text{all } \psi_j(\alpha_k) \\ j \neq k}}\left(\frac{1}{n_k(\alpha_j - \psi(\alpha_k))} - \frac{s_j^2}{n_k} - \frac{h'_{jj}}{n_j}\right)\right). \quad \text{(A.17)}$$

The resemblance to equation 4.55 in the ARD framework from section 4.2.1 is obvious but finding the maximum by setting it equal to zero and formulating an update equation where non-negativity of α_k is maintained is not straight forward here.

A.7 Fourier Transform

In chapter 6.1 the pre-processing of the data involves some spectral analysis methods where the signals are warped into the frequency domain. Here the definition of these methods are presented. For further details on these and related methods a book by *Mitra* [37] can be recommended.

Discrete Fourier Transform (DFT)

Let $x(n)$ be a discrete finite length sequence where $0 \leq n \leq N-1$. In the context of this thesis the signal $x(n)$ could represent a measured EEG channel voltage sampled in time. The discrete Fourier transform (DFT) $X_{DFT}(k)$ of the signal is then defined:

$$X_{DFT}(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}, \quad 0 \leq k \leq N-1 \tag{A.18}$$

where $X_{DFT}(k)$ is uniformly sampled on the ω-axis between $0 \leq \omega \leq 2\pi$ at $\omega = 2\pi k/N$. $X_{DFT}(k)$ is therefore a finite length sequence in the frequency domain of length N. Computer calculation of the DFT is generally obtained using an algorithm called *fast Fourier transform (FFT)*.

Spectrogram

The spectrogram, also know as the *short-time Fourier transform (STFT)*, of a finite length sequence $x(n)$ is defined

$$X_{STFT}(k,n) = \sum_{m=0}^{R-1} x(n-m)w(m)e^{-j2\pi km/N}, \quad 0 \leq k \leq N-1 \tag{A.19}$$

where $w(m)$ is a suitably chosen window function over the range $0 \leq m \leq R-1$ and $N \geq R$. The spectrogram $X_{STFT}(k,n)$ is therefore a function of two variables namely the discrete time indices n and discrete equally spaced frequencies $\omega_k = 2\pi k/N$.

Appendix B

Proposed Matlab Implementations

Here are listed the Matlab implementations of the algorithms from the thesis which were used for simulations and testing in chapters 5 and 6.

B.1 Algorithm I

```
function OUTPUT = algorithmI(m,A,epsilon,use_svd,maxIt,initVal,
    svdFile)
% OUTPUT = linearInverse(m,A,epsilon,use_svd,maxIt,initVal,svdFile)
% OUTPUT = linearInverse(m,A,epsilon,use_svd,maxIt,initVal)
% OUTPUT = linearInverse(m,A,epsilon,use_svd,maxIt)
% OUTPUT = linearInverse(m,A,epsilon,use_svd)
% OUTPUT = linearInverse(m,A,epsilon)
%
% INPUPS:
%   m (N x 1 vector) - measured signals
%   A (N x p matrix) - gain matrix
%   epsilon          - hyperparameter stop threshold
%   use_svd          - svd on/off
%   maxIt            - maximum number of iterations
%   initVal          - initial values of alpha and beta
%   svdFile          - file containing the SVD results for A
% OUTPUT: structure containing
%   alpha, beta - vectors showing convergence of hyperparameters
%   sMP         - vector with most probable s over iterations
%
% ----------------------------------------------------------------
% Author:  Thorsteinn Mar Arinbjarnarson
% Created: 1/3 2007
% Changes: 9/3 2007  - rewritten to reduce memory need
%          7/4 2007  - svd(A,'econ') used instead of svd(A)
%          4/7 2007  - alpha and beta intialized to 1
%          31/7 2007 - pre-allocate memory
%
% Algoritm I
% Based on the Bayesian formulation of the linear
% inverse problem from my MSc thesis.
% With use_svd=0 the equations are implemented "directly"
% so the algorithm crashes for large systems (out of memory).
% With use_svd=1 singular value decomposition is used
% along with some additional tricks to minimize memory
% usage and speed up the calculations.
% ----------------------------------------------------------------
if nargin < 4
    use_svd = 0;
    maxIt = 100;
elseif nargin < 5
    maxIt = 100; % default number of iterations
end

if nargin ~=6
    alpha = 1;%rand;
    beta =  1;%1e-1 * rand;
else
    if length(initVal)==2
        alpha = initVal(1);
        beta = initVal(2);
    else
        alpha = 1;%rand;
        beta =  1;%1e-1 * rand;
    end
end
stop = 0;
i = 1;
[N,p] = size(A);

% pre-allocate memory
alpha_old = zeros(1,maxIt+1);
beta_old = zeros(1,maxIt+1);
sMP = zeros(p,maxIt);

alpha_old(i) = alpha;
beta_old(i) = beta;
if use_svd
    tic
    disp('Starting SVD for alg. I...')
    AAT = A*A';
    eigATA = eig(AAT); % eig(A'*A) = eig(A*A')
    clear AAT
    if nargin == 7
        disp('..._loading_SVD_from_file_...')
        load(svdFile);
    else
        disp('..._running_SVD_algorithm_...')
        [U,D,V] = svd(A,'econ');
    end
    diagDTD = diag(D).^2;
    DTUTm = D'*U'*m;
    clear U D
    toc
    disp('Done!!!')
end

tic
disp('Starting_algorithm_I_iteration_loop...')
while(~stop)
    i = i + 1;
    if use_svd
        d = diagDTD + (alpha/beta);
        dInv = 1./d;
        clear d;
        idx = 1:length(dInv);
        dInv = sparse(idx,idx,dInv);
        clear idx;
        tmpVal = dInv*DTUTm;
        sMP(:,i-1) = V*tmpVal;
        clear dInv tmpVal
        lambda = beta*eigATA;
    else
        sMP(:,i-1) = inv(A'*A + alpha/beta * eye(p)) * A' * m;
        lambda = eig(beta*A'*A);
    end
    gma = sum(lambda./(lambda + alpha));
    Es = sum(sMP(:,i-1).^2)/2;
    Em = sum((m-A*sMP(:,i-1)).^2)/2;
    alpha = gma/(2*Es);
    beta = (N-gma)/(2*Em);
    alpha_old(i) = alpha;
    beta_old(i) = beta;
    diff = max([abs(alpha_old(i)-alpha_old(i-1)) ...
                abs(beta_old(i)-beta_old(i-1))]);
    if (diff<epsilon || i>maxIt) % stop criterion on
        hyperparameters
        stop = 1;
        toc
        disp('Done!!!')
    end
end

OUTPUT.alpha = alpha_old(:,1:i); clear alpha_old
OUTPUT.beta = beta_old(:,1:i);   clear beta_old
OUTPUT.sMP = sMP(:,1:i-1);       clear sMP
```

B.2 Algorithm II

```
function OUTPUT = algorithmII(m,A,epsilon,use_kv,maxIt,initVal)
% OUTPUT = algorithmII(m,A,epsilon,use_kv,maxIt,initVal)
% OUTPUT = algorithmII(m,A,epsilon,use_kv,maxIt)
% OUTPUT = algorithmII(m,A,epsilon,use_kv)
% OUTPUT = algorithmII(m,A,epsilon)
%
% INPUPS:
%   m (N x 1 vector) - measured signals
%   A (N x p matrix) - gain matrix
%   epsilon          - hyperparameter stop threshold, if negative
%                      use maxIt
%   use_kv           - Kailath Variant usage on/off
%   maxIt            - maximum number of iterations
%   initVal          - structure with initial values of alpha and
%                      beta (initVal.beta and initVal.beta)
% OUTPUT: structure containing
%   alpha, beta - vectors showing convergence of hyperparameters
%   sMP         - vector with most probable s over iterations
%
% -----------------------------------------------------------------
% Author  : Thorsteinn Mar Arinbjarnarson (MSc project - IMM/DTU)
% Created : 22/3 2007
% Modified: 29/3 2007 Kailath Variant used to improve algorithm
%           30/3 2007 beta update formula changed
%           3/4  2007 upper limit on beta set to 1e3*1/std(m)^2 and
%                     convergence criterion based on the min
%                     alpha value
%           27/4 2007 convergence criterion same as in algIIb,
%
% Algoritm II
% Based on the ARD Bayesian formulation of the linear
% inverse problem from my MSc thesis.
% -----------------------------------------------------------------
disp('Starting algorithm II...')
if nargin < 4
    use_kv = 0;
    maxIt = 100;
elseif nargin < 5
    maxIt = 100; % default number of iterations
end

[N,p] = size(A);

if nargin ~=6
    dAlpha = rand(p,1); % diagonal of the alpha matrix
    sBeta = 1e-1 * rand; % scalar beta value
else
    dAlpha = initVal.alpha;
    sBeta = initVal.beta;
end
stop = 0;
i = 1;
```

```
dAlpha_old(:,i) = dAlpha;
sBeta_old(i) = sBeta;

maxBeta = 1e3*1/std(m)^2;
maxAlpha = 10e6;

if ~use_kv
    AtA = A'*A;
end

disp('␣␣Starting␣Lambda␣and␣beta␣iteration␣loop...')
wh = waitbar(0,'Running␣algorithm');
while(~stop)
    i = i + 1;
    if use_kv
        idx = 1:length(dAlpha);
        dInv = 1./dAlpha;
        invLambda = sparse(idx,idx,dInv);
        invNN = inv(speye(N)./sBeta+A*invLambda*A');
        clear idx dInv
        T = (1/sBeta)*invLambda*A'*invNN;
        sMP(:,i-1)=sBeta*T*m;
        tr = sum(sum(sBeta*T.*A',2)); % trace
        clear T invLambda
        Em = sum((m-A*sMP(:,i-1)).^2)/2;
        sBeta = (N-tr)/(2*Em);
%         sBeta = N /(2*Em+tr*(1/sBeta));
        if sBeta > maxBeta
            sBeta = maxBeta;
        end
        clear Em tr
        hm_kk = calcHessDiag(A,invNN,dAlpha);
        clear invNN
        dAlpha = 1./(sMP(:,i-1).^2+hm_kk);
        clear hm_kk
    else
        invHess = inv(sBeta*AtA + sparse(diag(dAlpha)));
        sMP(:,i-1) = (sBeta*invHess) * A' * m;
        Em = sum((m-A*sMP(:,i-1)).^2)/2;
        %tr = trace(invHess*AtA);
        tr = trace(sBeta*invHess*AtA);
        sBeta = (N-tr)/(2*Em);
        hm_kk = diag(invHess);
        dAlpha = 1./(sMP(:,i-1).^2+hm_kk);
    end

    dAlpha_old(:,i) = dAlpha;
    sBeta_old(i) = sBeta;

    if ((i-2)-5 > 0) % at least 5 iterations!
        idx = find(dAlpha_old(:,i)<maxAlpha);
        diff = max( abs(dAlpha_old(idx,i)-dAlpha_old(idx,i-1)) );
        clear idx
    else
        diff = epsilon+1;
    end
    if epsilon<0
        stopCond = (i>maxIt);
    else
        stopCond = (diff<epsilon || i>maxIt(1));
    end
    if stopCond
        stop = 1;
        disp('Done!!!')
    end
    wbRefresh(i-1, maxIt(1),wh)
end

OUTPUT.alpha = dAlpha_old;
OUTPUT.beta = sBeta_old;
OUTPUT.sMP = sMP;
close(wh)

% Sub-function to refresh waitbar
function wbRefresh(it, max, handle)
set(handle,'Name',[num2str(it) ' iterations of max ' num2str(max)])
ratio = it/max;
waitbar(ratio,handle)

% Sub-function to calculate only the diagonal
% elements of the hessian inverse matrix.
%
% Inputs:
%  A     - Gain matrix (N x p)
%  invHH - matrix  (N x N), i.e. inv(1/beta*I+A*invLambda*A')
%  diagL - diagonal elements of Lambda (p x 1)
function h = calcHessDiag(AA,NN,diagL)
h = sum(AA'*NN.*AA',2);
h = (1./diagL) .* (1 - h./diagL);
```

B.3 Algorithm IIb

```
function OUTPUT = algorithmIIb(m,A,epsilon,use_kv,maxIt,mag)
% OUTPUT = algorithmIIb(m,A,epsilon,use_kv,maxIt,mag)
% OUTPUT = algorithmIIb(m,A,epsilon,use_kv,maxIt)
% OUTPUT = algorithmIIb(m,A,epsilon,use_kv)
% OUTPUT = algorithmIIb(m,A,epsilon)
%
% INPUPS:
%   m (N x 1 vector) - measured signals
%   A (N x p matrix) - gain matrix
%   epsilon          - hyperparameter stop threshold, epsilon(1)
%                      for algI/beta and epsilon(2) for
%                      Lambda/algII
%                      if epsilon(2)<0 use maximum iterations
%   use_kv           - Kailath Variant usage on/off
%   maxIt            - maximum number of iterations,
%                      can be a vector
%                      specifying for algI also
%   mag              - multiplication parameter for the max
%                      alpha value
% OUTPUT: structure containing
%   alpha, beta - vectors showing convergence of hyperparameters
%   sMP         - vector with most probable s over iterations
%
% ------------------------------------------------------------------
% Author   : Thorsteinn Mar Arinbjarnarson (MSc project - IMM/DTU)
% Created  : 2/4 2007
% Modified: 23/4 2007 max values on alpha and beta,
%                     convergence criterion changed
%           26/4 2007 subfunction h = calcHessDiag(AA,NN,diagL)
%                     rewritten
%           3/7 2007  'initVal' and 'svdFile' inputs removed and
%                     'mag' input added instead
%
% Algoritm IIb
% Based on the ARD formulation of the linear
% inverse problem from my MSc thesis.
% Uses algortihm I to find beta, then using this beta
% values goes on to use algorithm II to find the
% diagonal matrix of alpha values.
% Note that the number of iterations for beta and alpha
% are therefore not necessarily the same.
% ------------------------------------------------------------------
disp('Starting algorithm IIb...')
if nargin < 4
    use_kv = 0;
    maxIt = 100;
elseif nargin < 5
    maxIt = 100; % default number of iterations
end

[N,p] = size(A);

```

```
if nargin < 6
    mag = 100; % default value
end

if length(epsilon) > 1
    epsAlgI = epsilon(1);
    epsAlgII = epsilon(2);
elseif length(epsilon) == 1
    epsAlgI = epsilon;
    epsAlgII = epsilon;
end

dAlpha = rand(p,1); % diagonal of the Lambda matrix init
algI_out = algorithmI(m,A,epsAlgI,1,maxIt(1));
sBeta = algI_out.beta(end); % scalar beta value
maxAlpha = algI_out.alpha(end)*mag; % Set max alpha value ("
    infinity")

if length(maxIt)==2
    maxIt(1) = maxIt(2);
end

% max sBeta criterion, same as is used in algorithm II
if sBeta > 1e3*1/std(m)^2
    sBeta = 1e3*1/std(m)^2;
    algI_out.beta(end) = sBeta;
end
OUTPUT.beta = algI_out.beta;

clear epsAlgI algI_out
stop = 0;
i = 1;
dAlpha_old(:,i) = dAlpha;

if ~use_kv
    AtA = A'*A;
end

tic
disp('  Starting Lambda iteration loop...')
wh = waitbar(0,'  Running algorithm');
while(~stop)
    i = i + 1;
    if use_kv
        idx = 1:length(dAlpha);
        dInv = 1./dAlpha;
        invLambda = sparse(idx,idx,dInv);
        invNN = inv(speye(N)./sBeta+A*invLambda*A');
        clear idx dInv
        T = (1/sBeta)*invLambda*A'*invNN;
        sMP(:,i-1)=sBeta*T*m;
        clear T invLambda
        hm_kk = calcHessDiag(A,invNN,dAlpha);
        clear invNN
        dAlpha = 1./(sMP(:,i-1).^2+hm_kk);
        clear hm_kk

        % Check if max alpha value exceeded
        idx = find(dAlpha>maxAlpha);
        dAlpha(idx) = maxAlpha;
        clear idx
    else
        invHess = inv(sBeta*AtA + sparse(diag(dAlpha)));
        sMP(:,i-1) = (sBeta*invHess) * A' * m;
        hm_kk = diag(invHess);
        dAlpha = 1./(sMP(:,i-1).^2+hm_kk);
    end

    dAlpha_old(:,i) = dAlpha;

    if ((i-2)-5 > 0) % at least 5 iterations!
        idx = find(dAlpha_old(:,i)<maxAlpha);
        diff = max( abs(dAlpha_old(idx,i)-dAlpha_old(idx,i-1)) );
        clear idx
    else
        diff = epsAlgII+1;
    end
    if epsAlgII<0
        stopCond = (i>maxIt(1));
    else
        stopCond = (diff<epsAlgII || i>maxIt(1));
    end
    if stopCond
        stop = 1;
        toc
        disp('Done!!!')
    end
    wbRefresh(i-1, maxIt(1),wh)
end

OUTPUT.alpha = dAlpha_old;
OUTPUT.sMP = sMP;
close(wh)

% Sub-function to refresh waitbar
function wbRefresh(it, max, handle)
set(handle,'Name',[num2str(it) ' iterations of max ' num2str(max)])
ratio = it/max;
waitbar(ratio,handle)

% Sub-function to calculate only the diagonal
% elements of the hessian inverse matrix.
%
% Inputs:
%  A     - Gain matrix (Nxp)
%  invHH - matrix  (NxN), i.e. inv(1/beta*I+A*invLambda*A')
%  diagL - diagonal elements of Lambda (px1)
function h = calcHessDiag(AA,NN,diagL)
h = sum(AA'*NN.*AA',2); % tkk
h = (1./diagL) .* (1 - h./diagL);
```

B.4 Algorithm IIc

```
function OUTPUT = algorithmIIc(m,A,epsilon,use_kv,maxIt,mag)
% OUTPUT = algorithmIIc(m,A,epsilon,use_kv,maxIt,mag)
% OUTPUT = algorithmIIc(m,A,epsilon,use_kv,maxIt)
% OUTPUT = algorithmIIc(m,A,epsilon,use_kv)
% OUTPUT = algorithmIIc(m,A,epsilon)
%
% INPUPS:
%   m (N x 1 vector) - measured signals
%   A (N x p matrix) - gain matrix
%   epsilon          - hyperparameter stop threshold, epsilon(1)
%                      for algI/beta and epsilon(2) for
%                      Lambda/algII
%                      if epsilon(2)<0 use maximum iterations
%   use_kv           - Kailath Variant usage on/off
%                      NOTE: use_kv == 0 no longer supported
%   maxIt            - maximum number of iterations, can be a
%                      vector specifying for algI also
%   mag              - multiplication parameter for the max
%                      alpha value
% OUTPUT: structure containing
%   alpha, beta - vectors showing convergence of hyperparameters
%   sMP         - vector with most probable s over iterations
%   pv          - vector with active sets over iterations
%
% -------------------------------------------------------------------
% Author   : Thorsteinn Mar Arinbjarnarson (MSc project - IMM/DTU)
% Created  : 1/5 2007
% Modified: 10/5 2007 - SVD and initVal inputs removed and
%                       mag input, which
%                       scales the maxAlpha value, added.
%
% Algoritm IIc
% Same as algorithm IIb but active set feature added,
% meaning that inactive sets, where alpha_k
% has reached the maximum value, are removed from the model
% speeding up the calculations.
% -------------------------------------------------------------------

disp('Starting algorithm IIc...')
if nargin < 4
    use_kv = 0;
    maxIt = 100;
elseif nargin < 5
    maxIt = 100; % default number of iterations
end

[N,p] = size(A);
```

```
p_true = p;

if nargin < 6
    mag = 100; % default value
end

if length(epsilon) > 1
    epsAlgI = epsilon(1);
    epsAlgII = epsilon(2);
elseif length(epsilon) == 1
    epsAlgI = epsilon;
    epsAlgII = epsilon;
end

dAlpha = rand(p,1); % diagonal of the Lambda matrix init
%dAlpha = ones(p,1);
algI_out = algorithmI(m,A,epsAlgI,1,maxIt(1));
sBeta = algI_out.beta(end); % scalar beta value
maxAlpha = algI_out.alpha(end)*mag; % Set max alpha value ("
    infinity")

beta_out = algI_out.beta;
clear algI_out

if length(maxIt)==2
    maxIt(1) = maxIt(2);
end

% max sBeta criterion, same as is used in algorithm II
if sBeta > 1e3*1/std(m)^2
    sBeta = 1e3*1/std(m)^2;
end

clear epsAlgI mag
stop = 0;
i = 1;
dAlphaE(:,i) = dAlpha;
pv(i) = p;

% if ~use_kv % Not supported anymore
%     AtA = A'*A;
% end

tic
disp('  Starting Lambda iteration loop...')
wh = waitbar(0,'  Running algorithm');
active = 1:p; % vector that keeps track of active classes
while(~stop)
    i = i + 1;
    if 1%use_kv
        idx = 1:length(dAlpha);
        dInv = 1./dAlpha;
        invLambda = sparse(idx,idx,dInv);
        invNN = inv(speye(N)./sBeta+A*invLambda*A');
        clear idx dInv
        T = (1/sBeta)*invLambda*A'*invNN;
        sMP=sBeta*T*m;
        clear T invLambda
        hm_kk = calcHessDiag(A,invNN,dAlpha);
        clear invNN
        dAlpha = 1./(sMP.^2+hm_kk);
        clear hm_kk

        % Check if max alpha value exceeded
        k = find(dAlpha>=maxAlpha);
        dAlpha(k) = maxAlpha;
        if length(k)>0
            % Romove inactive sets.
            % First some book keeping
            for j=1:length(k)
                idx = find(active==k(j));
                active(idx) = 0;
            end
            idx = find(active==0);
            for j=1:length(idx)
                active(idx(j):end) = active(idx(j):end)-1;
            end
            % Now remove sources from model
            % (i.e. reduce A, s and dAlpha)
            idx = 1:k(1)-1;
            for j=1:length(k)-1 % create index vector for the
                indices to keep
                idx = [idx k(j)+1:k(j+1)-1];
            end
            idx = [idx k(end)+1:p];
            % Remove from model
            sMP = sMP(idx);
            dAlpha = dAlpha(idx);
            A = A(:,idx);
            p = p-length(k);
        end
        % Expand model again for storing
        idx = find(active>0);
        sMPe(:,i-1) = zeros(p_true,1);
        sMPe(idx,i-1) = sMP;
        dAlphaE(:,i) = repmat(maxAlpha,p_true,1);
        dAlphaE(idx,i) = dAlpha;
        pv(i) = p;
        clear k idx
    else % Not supported anymore!
        invHess = inv(sBeta*AtA + sparse(diag(dAlpha)));
        sMP(:,i-1) = (sBeta*invHess) * A' * m;
        hm_kk = diag(invHess);
        dAlpha = 1./(sMP(:,i-1).^2+hm_kk);
    end

    if ((i-2)-5 > 0) % at least 5 iterations!
        idx = find(dAlphaE(:,i)<maxAlpha);
        diff = max( abs(dAlphaE(idx,i)-dAlphaE(idx,i-1)) );
        clear idx
    else
        diff = epsAlgII+1;
    end
    if epsAlgII<0
        stopCond = (i>maxIt(1));
    else
        stopCond = (diff<epsAlgII || i>maxIt(1));
    end
    if stopCond
        stop = 1;
        toc
        disp('Done!!!')
    end
    wbRefresh(i-1, maxIt(1),wh)
end

OUTPUT.alpha = dAlphaE;
OUTPUT.beta = beta_out;
OUTPUT.sMP = sMPe;
OUTPUT.pv = pv;
close(wh)

% Sub-function to refresh waitbar
function wbRefresh(it, max, handle)
set(handle,'Name',[num2str(it) ' iterations of max ' num2str(max)])
ratio = it/max;
waitbar(ratio,handle)

% Sub-function to calculate only the diagonal
% elements of the hessian inverse matrix.
%
% Inputs:
%  A     - Gain matrix (Nxp)
%  invHH - matrix  (NxN), i.e. inv(1/beta*I+A*invLambda*A')
%  diagL - diagonal elements of Lambda (px1)
function h = calcHessDiag(AA,NN,diagL)
h = sum(AA'*NN.*AA',2); % tkk
h = (1./diagL) .* (1 - h./diagL);
```

B.5 Algorithm IId

```
function OUTPUT = algorithmIId(m,A,epsilon,maxIt,con,q,mag,irrIdcs)
% OUTPUT = algorithmIId(m,A,epsilon,maxIt,con,q,mag,irrIdcs)
% OUTPUT = algorithmIId(m,A,epsilon,maxIt,con,q,mag)
%
% INPUPS:
%   m (N x 1 vector) - EEG signals
%   A (N x p matrix) - gain matrix (lead field matrix)
%   epsilon          - hyperparameter stop threshold, epsilon(1)
%                      for algI/beta and epsilon(2) for
%                      Lambda/algII if epsilon(2)<0 use maximum
%                      iterations
%   maxIt            - maximum number of iterations,
%                      can be a vector
%                      specifying for algI also
%   con              - connectivity of sources, used for low pass
%                      filtering
%   q                - q factor for avg filter
%   mag              - multiplication factor for max alpha value
%   irrIdcs          - indices to non-relevant alpha's which are
%                      forced to infinity
% OUTPUT: structure containing
%   dAlpha, beta - matrix and vector showing convergence of
%                  hyperparameters
%   sMP          - matrix with most probable s over iterations
%   pv           - vector with active sets over iterations
%
% ------------------------------------------------------------
% Author   : Thorsteinn Mar Arinbjarnarson (MSc project - IMM/DTU)
% Created  : 15/5 2007
% Modified: 28/6 2007 'q' added to inputs and connectivity support
%                     for 2D tests
%           13/7 2007 'irrIdcs' input added, specifies indices to
%                     non-relevant alpha's that are set to infinity
%           27/7 2007 use active sets to reduce computation time
%
% Algoritm IId
% Based on algorithm IIb with LP filtering on sMP during
% iteration (averaging window).
% ------------------------------------------------------------
disp('Starting algorithm IId...')

[N,p] = size(A);
p_true = p;

if nargin < 7
    mag = 100; % default value
end

if length(epsilon) > 1
    epsAlgI = epsilon(1);
    epsAlgII = epsilon(2);
elseif length(epsilon) == 1
```

```
    epsAlgI = epsilon;
    epsAlgII = epsilon;
end

dAlpha = rand(p,1); % diagonal of the Lambda matrix init
algI_out = algorithmI(m,A,epsAlgI,1,maxIt(1));
sBeta = algI_out.beta(end); % scalar beta value
maxAlpha = algI_out.alpha(end)*mag; % Set max alpha value ("
    infinity")

if length(maxIt)==2
    maxIt(1) = maxIt(2);
end

% max sBeta criterion, same as is used in algorithm II
if sBeta > 1e3*1/std(m)^2
    sBeta = 1e3*1/std(m)^2;
    OUTPUT.beta = sBeta;
else
    OUTPUT.beta = algI_out.beta;
end

% pre-allocate memory
dAlphaE = zeros(p,maxIt(2)+1);
sMPe = zeros(p,maxIt(2));
pv = zeros(maxIt(2)+1,1);

clear epsAlgI
stop = 0;
i = 1;
dAlphaE(:,i) = dAlpha;
pv(i) = p;
idxFull = 1:p;

disp('Starting␣algIId␣iteration␣loop...')
wh = waitbar(0,'␣␣Running␣algorithm');
active = 1:p; % vector that keeps track of active classes
while(~stop)
    i = i + 1;
    idx = 1:length(dAlpha);
    dInv = 1./dAlpha;
    invLambda = sparse(idx,idx,dInv);
    invNN = inv(speye(N)./sBeta+A*invLambda*A');
    clear idx dInv
    T = (1/sBeta)*invLambda*A'*invNN;
    sMP=sBeta*T*m;

    % Low pass filtering over whole surface
    sMPfull = zeros(p_true,1);
    sMPfull(idxFull) = sMP;
    sMPfull = lpFilt(sMPfull,con,q);
    sMP = sMPfull(idxFull,1);

    clear T invLambda
    hm_kk = calcHessDiag(A,invNN,dAlpha);
    clear invNN
    dAlpha = 1./(sMP.^2+hm_kk);
    clear hm_kk

    % force out some selected sources in first iteration run
    if nargin==8 && i==2
        dAlpha(irrIdcs) = maxAlpha+1;
    end
    % Check if max alpha value exceeded
    k = find(dAlpha>=maxAlpha);
    dAlpha(k) = maxAlpha;

    if length(k)>0 % Romove inactive sets.
        % First some book keeping
        for j=1:length(k)
            idx = find(active==k(j));
            active(idx) = 0;
        end
        idx = find(active==0);
        for j=1:length(idx)
            active(idx(j):end) = active(idx(j):end)-1;
        end
        % Now remove sources from model
        % (i.e. reduce A, s and dAlpha)
        idx = 1:k(1)-1;
        for j=1:length(k)-1 % create index vector for the indices
            to keep
            idx = [idx k(j)+1:k(j+1)-1];
        end
        idx = [idx k(end)+1:p];
        % Finally remove from model
        sMP = sMP(idx);
        dAlpha = dAlpha(idx);
        A = A(:,idx);
        p = p-length(k);
    end

    % Expand model again for storing
    idxFull = find(active>0);
    sMPe(:,i-1) = zeros(p_true,1);
    sMPe(idxFull,i-1) = sMP;
    dAlphaE(:,i) = repmat(maxAlpha,p_true,1);
    dAlphaE(idxFull,i) = dAlpha;
    pv(i) = p;
    clear k idx

    if ((i-2)-5 > 0) % at least 5 iterations!
        idx = find(dAlphaE(:,i)<maxAlpha);
        diff = max( abs(dAlphaE(idx,i)-dAlphaE(idx,i-1)) );
        clear idx
    else
        diff = epsAlgII+1;
    end
    if epsAlgII<0
        stopCond = (i>maxIt(1));
    else
        stopCond = (diff<epsAlgII || i>maxIt(1));
    end
    if stopCond
        stop = 1;
        toc
        disp('Done!!!')
    end
    wbRefresh(i-1, maxIt(1),wh)
end

OUTPUT.dAlpha = dAlphaE(:,1:i); clear dAlphaE;
OUTPUT.sMP = sMPe(:,1:i-1); clear sMPe;
OUTPUT.pv = pv(1:i);
close(wh)

% Sub-function to refresh waitbar
function wbRefresh(it, max, handle)
set(handle,'Name',[num2str(it) ' iterations of max ' num2str(max)])
ratio = it/max;
waitbar(ratio,handle)

% Sub-function to calculate only the diagonal
% elements of the hessian inverse matrix.
%
% Inputs:
%  A     - Gain matrix (Nxp)
%  invHH - matrix  (NxN), i.e. inv(1/beta*I+A*invLambda*A')
%  diagL - diagonal elements of Lambda (px1)
function h = calcHessDiag(AA,NN,diagL)
h = sum(AA'*NN.*AA',2); % tkk
h = (1./diagL) .* (1 - h./diagL);

% Low pass filter sMP using an averaging window
% s_in     - sMP input
% con_in   - connectivity cell array
% q_in     - q factor for avg window
% s_out    - lp filterted sMP
function s_out = lpFilt(s_in,con_in,q_in)
s_out = s_in;
for m=1:length(con_in)
    s_out(m) = q_in*s_in(m) + ...
               ((1-q_in)/length(s_in(con_in{m})))* sum(s_in(con_in{
                   m}));
end
```

B.6 Algorithm III

```
function OUTPUT = algorithmIII(m,A,epsilon,maxIt,mag,vertconn)
% OUTPUT = algorithmIII(m,A,epsilon,maxIt,mag)
% OUTPUT = algorithmIII(m,A,epsilon,maxIt)
% OUTPUT = algorithmIII(m,A,epsilon)
%
% INPUPS:
%   m (N x 1 vector) - measured signals
%   A (N x p matrix) - gain matrix
%   epsilon          - hyperparameter stop threshold,
%                      if negative use maxIt
%   maxIt            - maximum number of iterations
%   mag              - multiplication constant to set maxAlpha
%                      value (typical val. beween 0.1 and 100)
%   vertconn         - connectivity of the sources
% OUTPUT: structure containing
%   beta, dAlpha - vectors showing convergence of hyperparameters
%   sMP          - vector with most probable s over iterations
%
% -----------------------------------------------------------------
% Author  : Thorsteinn Mar Arinbjarnarson (MSc project - IMM/DTU)
% Created : 27/6 2007
%
% Algoritm III
% Spatially smoothing prior to estimate sMP and same update
% equations as in algorithm IIb to estimate hyperparameters.
% -----------------------------------------------------------------
disp('Starting algorithm III...')
if nargin < 4
    maxIt = 100;
end
[N,p] = size(A);
if nargin < 5
    mag = 100;
end
if length(epsilon) > 1
    epsAlgI = epsilon(1);
    epsAlgII = epsilon(2);
elseif length(epsilon) == 1
    epsAlgI = epsilon;
    epsAlgII = epsilon;
end

% initalize hyperparameters alpha_k
dAlpha = ones(p,1);
algI_out = linearInverse(m,A,epsAlgI,1,maxIt(1));
sBeta = algI_out.beta(end);
maxAlpha = algI_out.alpha(end)*mag; % Set max alpha value ("
    infinity")

if length(maxIt)==2
    maxIt(1) = maxIt(2);
end
```

```

% max sBeta criterion, same as is used in algorithm II
if sBeta > 1e3*1/std(m)^2
    sBeta = 1e3*1/std(m)^2;
end

OUTPUT.beta = algI_out.beta;
clear epsAlgI algI_out

stop = 0;
i = 1;
dAlpha_old(:,i) = dAlpha;

wh = waitbar(0,'Running␣algorithm');
while(~stop)
    i = i + 1;
    idx = 1:length(dAlpha);
    psi = calcPsi(vertconn,dAlpha);
    dInv = 1./(dAlpha+psi);
    invGamma = sparse(idx,idx,dInv);
    invNN = inv(speye(N)./sBeta+A*invGamma*A');
    clear dInv
    T = (1/sBeta)*invGamma*A'*invNN;
    sMP = sBeta*T*m;
    clear T invGamma invNN
    dInv = 1./(dAlpha);
    invLambda = sparse(idx,idx,dInv);
    invNN = inv(speye(N)./sBeta+A*invLambda*A');
    clear invLambda
    hm_kk = calcHessDiag(A,invNN,dAlpha);
    clear invNN idx
    dAlpha = 1 ./ (sMP.^2+hm_kk);
    clear hm_kk
    idx = find(dAlpha>maxAlpha);
    dAlpha(idx) = maxAlpha;
    clear idx

    sMPm(:,i-1) = sMP;
    dAlpha_old(:,i) = dAlpha;

    if ((i-2)-5 > 0) % at least 5 iterations!
        idx = find(dAlpha_old(:,i)<maxAlpha);
        diff = max( abs(dAlpha_old(idx,i)-dAlpha_old(idx,i-1)) );
        clear idx
    else
        diff = epsAlgII+1;
    end
    if epsAlgII<0
        stopCond = (i>maxIt);
    else
        stopCond = (diff<epsAlgII || i>maxIt(1));
    end
    if stopCond
        stop = 1;
        disp('Done!!!')
    end
    wbRefresh(i-1, maxIt(1),wh)
end

OUTPUT.dAlpha = dAlpha_old;
OUTPUT.sMP = sMPm;
close(wh)

% Sub-function to refresh waitbar
function wbRefresh(it, max, handle)
set(handle,'Name',[num2str(it) ' iterations of max ' num2str(max)])
ratio = it/max;
waitbar(ratio,handle)

% Sub-function to calculate only the diagonal
% elements of the hessian inverse matrix.
%
% Inputs:
%  A     - Gain matrix (Nxp)
%  invHH - matrix  (NxN), i.e. inv(1/beta*I+A*invLambda*A')
%  diagL - diagonal elements of Lambda (px1)
function h = calcHessDiag(AA,NN,diagL)
h = sum(AA'*NN.*AA',2);
h = (1./diagL) .* (1 - h./diagL);

% Sub-function to calculate the psi parameters.
%
% Inputs:
% con - connectivity of source tessellation surface
% alphas - alpha_k values (dAlpha)
function cPsi = calcPsi(con,alphas)
l = length(con);
cPsi = zeros(l,1);
for m=1:length(con)
    j = con{m};
    alpha_p = alphas(j);
    n_p = zeros(length(j),1);
    for k=1:length(j)
        n_p(k) = length(con{j(k)});
    end
    cPsi(m) = sum(alpha_p./n_p);
end
clear l n_p alpha_p j m
```

Appendix C

BEM Head Model Details

The main focus of the thesis are EEG inverse methods but the forward model plays a role in the inverse calculations. For the sake of reproducibility this appendix provides details on BEM head models used in the simulations and testing chapters.

C.1 256 Channel BEM

Tesselation surfaces for an imaginary subject are generated using a high density 256 channel EEG electrode grid supplied with the BrainStorm software. And as described in section 3.5 the Montreal Phantom head is warped to match the channel locations. The sources are localized on the cortical surface with directions perpendicular to the surface. Number of EEG channels is $N = 256$ and the number of discretized source locations on the cortex surface is $P = 10001$, the resulting lead field matrix $\mathbf{A}$ therefore has dimensions 256×10001. The warped phantom surfaces used for the BEM modeling are the scalp, outer skull, inner skull (or equivalently CSF) and finally the cortex to place the sources on. Linear collocations BEM method is used. Other setup information details are listed in the following Matlab scrip. Some minor modifications were done to the BrainStorm function that does the actual forward model calculations (bst_headmodeler.m) regarding naming of files and path locations of saved files. The algorithm itself was not modified.

BEMheadModel.m

```
% function BEMheadModel
% -----------------------------------------------------------------
% Author :  Thorsteinn Mar Arinbjarnarson (DTU/IMM - MSc project)
% Created:  12/2 2007
% Modified: 16/4 2007 - use BrainStorm DB structure
%
% Script that creates a head model using BEM
% (boundary element method) for the BrainStorm EGI demo data.
% -----------------------------------------------------------------
% BrainStorm m-files used:
% - headmodeler_gui.m
% - bst_headmodeler.m
% -----------------------------------------------------------------
[userDB idx] = setDBpath();

OPTIONS.HeadModelFile = [userDB(idx).STUDIES '/EGIdemo_BEM'];
OPTIONS.ImageGridFile = 'Default';
OPTIONS.Method = {'eeg_bem'}; % method: BEM
OPTIONS.HeadModelName = 'BEM';
OPTIONS.Verbose = 1;
OPTIONS.Scalp.FileName = [userDB(idx).SUBJECTS '/EGIdemo_warped_4layer_tess.mat'];
OPTIONS.Scalp.iGrid = 3; % Head Compartments; Scalp: warpedScalp
OPTIONS.Cortex.FileName = [userDB(idx).SUBJECTS '/EGIdemo_warped_4layer_tess.mat'];
OPTIONS.Cortex.iGrid = 4;% Head Compartments; Cortex: warpedWhiteMatter
OPTIONS.EEGRef = 'Cz';              % EEG reference: Cz
OPTIONS.BEM.Interpolative = 0;     % Non-Interpolative or interpolative
OPTIONS.BEM.checksurf = 1;         % Align surfaces: On/Off
OPTIONS.BEM.Basis = 'linear';      % 'linear' or 'constant'
OPTIONS.BEM.Test = 'Collocation';  % Test: 'Collocation' or 'Galerkin'
OPTIONS.BEM.NVertMax = 1000;       % Decimate envelopes down to
OPTIONS.BEM.ISA = 1;               % Insulated skull approach: On
OPTIONS.BEM.ForceXferComputation = 0; % Recompute trancfer matrices Yes/No
OPTIONS.BEM.EnvelopeNames{1}.TessName = 'warpedCSF'; % ordered envelopes: warpedCSF, warpedSkull, warpedScalp
OPTIONS.BEM.EnvelopeNames{1}.TessFile = [userDB(idx).SUBJECTS '/EGIdemo_warped_4layer_tess'];
OPTIONS.BEM.EnvelopeNames{2}.TessName = 'warpedskull';
OPTIONS.BEM.EnvelopeNames{2}.TessFile = [userDB(idx).SUBJECTS '/EGIdemo_warped_4layer_tess'];
OPTIONS.BEM.EnvelopeNames{3}.TessName = 'warpedscalp';
OPTIONS.BEM.EnvelopeNames{3}.TessFile = [userDB(idx).SUBJECTS '/
    EGIdemo_warped_4layer_tess'];
OPTIONS.SourceModel = -1;      % Source model: Current dipole
OPTIONS.VolumeSourceGrid = 0; % Volume source grid: on/off
OPTIONS.VolumeSourceGridSpacing = 2; % 2cm is the default value
OPTIONS.ChannelFile = [userDB(idx).STUDIES '/EGIdemo_channel.mat'];
OPTIONS.ChannelType = 'EEG';

% OPTIONS.TessFileComment is a cell array of length ntess (number
% of tess files). kth cell contains a nsurf (number of surfaces in
% kth tess file) by two array. First column is the index of the
% tess file in OPTIONS.TessellationFile and second column is the
% index of the surface in the Comment field of the Tess file.
OPTIONS.TessFileComment{1} = [1 1; 1 2; 1 3; 1 4];
OPTIONS.TessFileComment{2} = [2 1; 2 2; 2 3; 2 4];
OPTIONS.MEGMethods = {'meg_sphere' 'meg_os' 'meg_bem'};
OPTIONS.MEGMethodsLabel = {'Single Sphere' 'Overlapping Spheres' '
    BEM'};
OPTIONS.EEGMethods = {'eeg_sphere' 'eeg_3sphere' 'eeg_3sphereBerg
    ' 'eeg_os' 'eeg_bem'};
OPTIONS.EEGMethodsLabel = {'Single Sphere' '3-shell Sphere' '3-
    shell Sphere (BERG)' 'Overlapping Spheres' 'BEM'};

[G,OPTIONS2] = bst_headmodeler_mod(OPTIONS);
movefile('*.bin', userDB(idx).STUDIES) % move all data files to the
     same folder
```

C.2 118 Channel BEM

Here the details are listed for the BEM head model used in section 6.3 for analysis of the BCI data. As noted in the section just referenced there are 118 channels and the tesselation surfaces for the head were generated from the Montreal Brain Phantom. The sources are localized on the cortical surface with directions perpendicular to the surface. Number of EEG channels is $N = 118$ and the number of discretized source locations on the cortex surface is $P = 10001$, the resulting lead field matrix $\mathbf{A}$ therefore has dimensions 118×10001. The warped phantom surfaces used for the BEM modeling are the scalp, outer skull, inner skull (or equivalently CSF) and finally the cortex to place the sources on. Linear collocations BEM method is used and other setup information is listed in the following Matlab scrip. Some minor modifications were done to the BrainStorm function that does the actual forward model calculations (bst_headmodeler.m) regarding naming of files and path locations of saved files. The algorithm itself was not modified.

BEMheadModelBCI.m

```
% function BEMheadModelBCI
% ----------------------------------------------------------------
% Author :  Thorsteinn Mar Arinbjarnarson (DTU/IMM - MSc project)
% Created:  9/7 2007
%
% Build a BEM head model for the BCI dataset.
% This script does the same as 'BEMheadModel.m' does for the
% EGIdemo dataset.
% ----------------------------------------------------------------
dbName = 'BCIdata';
[userDB idx] = setDBpath(dbName);

OPTIONS.HeadModelFile = [userDB(idx).STUDIES '/BCIdata_BEM'];
OPTIONS.ImageGridFile = 'Default';
OPTIONS.Method = {'eeg_bem'}; % method: BEM
OPTIONS.HeadModelName = 'BEM';
OPTIONS.Verbose = 1;
OPTIONS.Scalp.FileName = [userDB(idx).SUBJECTS '/
    BCI_warped_4layer_tess.mat'];
OPTIONS.Scalp.iGrid = 3; % Head Compartments; Scalp: warpedScalp
OPTIONS.Cortex.FileName = [userDB(idx).SUBJECTS '/
    BCI_warped_4layer_tess.mat'];
OPTIONS.Cortex.iGrid = 4;% Head Compartments; Cortex:
    warpedWhiteMatter
%OPTIONS.EEGRef = 'Cz';            % avg ref used, not Cz
OPTIONS.BEM.Interpolative = 0;     % Non-Interpolative or
    interpolative
OPTIONS.BEM.checksurf = 1;         % Align surfaces: On/Off
OPTIONS.BEM.Basis = 'linear';      % 'linear' or 'constant'
OPTIONS.BEM.Test = 'Collocation'; % Test: 'Collocation' or '
    Galerkin'
OPTIONS.BEM.NVertMax = 1000;       % Decimate envelopes down to
OPTIONS.BEM.ISA = 1;               % Insulated skull approach: On
OPTIONS.BEM.ForceXferComputation = 0; % Recompute trancfer matrices
     Yes/No
OPTIONS.BEM.EnvelopeNames{1}.TessName = 'warpedCSF'; % ordered
    envelopes: warpedCSF, warpedSkull, warpedScalp
OPTIONS.BEM.EnvelopeNames{1}.TessFile = [userDB(idx).SUBJECTS '/
    BCI_warped_4layer_tess'];
OPTIONS.BEM.EnvelopeNames{2}.TessName = 'warpedskull';
OPTIONS.BEM.EnvelopeNames{2}.TessFile = [userDB(idx).SUBJECTS '/
    BCI_warped_4layer_tess'];
OPTIONS.BEM.EnvelopeNames{3}.TessName = 'warpedscalp';
OPTIONS.BEM.EnvelopeNames{3}.TessFile = [userDB(idx).SUBJECTS '/
    BCI_warped_4layer_tess'];
OPTIONS.SourceModel = -1;      % Source model: Current dipole
OPTIONS.VolumeSourceGrid = 0; % Volume source grid: on/off
OPTIONS.ChannelFile = [userDB(idx).STUDIES '/BCI_channel.mat'];
OPTIONS.ChannelType = 'EEG';

% OPTIONS.TessFileComment is a cell array of length ntess (number
% of tess files). kth cell contains a nsurf (number of surfaces in
% kth tess file) by two array. First column is the index of the
% tess file in OPTIONS.TessellationFile and second column is the
% index of the surface in the Comment field of the Tess file.
OPTIONS.TessFileComment{1} = [1 1; 1 2; 1 3; 1 4];
OPTIONS.TessFileComment{2} = [2 1; 2 2; 2 3; 2 4];
OPTIONS.MEGMethods = {'meg_sphere' 'meg_os' 'meg_bem'};
OPTIONS.MEGMethodsLabel = {'Single␣Sphere' 'Overlapping␣Spheres' '
    BEM'};
OPTIONS.EEGMethods = {'eeg_sphere' 'eeg_3sphere' 'eeg_3sphereBerg
    ' 'eeg_os' 'eeg_bem'};
OPTIONS.EEGMethodsLabel = {'Single␣Sphere' '3-shell␣Sphere' '3-
    shell␣Sphere␣(BERG)' 'Overlapping␣Spheres' 'BEM'};

[G,OPTIONS2] = bst_headmodeler_mod(OPTIONS);
movefile('*.bin', userDB(idx).STUDIES) % move all data files to the
    same folder
```

Appendix D

High Density EEG Data Analysis

In the BEM simulations in chapter 5 a head model was created from the Montreal Brain Phantom using a 256 channel EEG grid from a subject. This channel grid was supplied with the BrainStorm software. Some sample EEG recordings are also available for these channels. Very little information is however supplied with the data making it hard for one to evaluate the quality of inverse calculations. Here this data will be analyzed using Algorithms I and IId. The main analysis will be with regards to the algorithms, i.e. do they converge nicely and does Algorithm IId give a more sparse representation of the Algorithm I estimate. The data is a single epoch of an event related potential, some event suggestions will therefore be put forth in the end. This can however not be validated since no such information for the data is available.

D.1 Results

Forward modeling for the subject is described in appendix C.1. The lead field matrix **A** is of dimensions 256×10001 and very ill-conditioned. On figure D.1 all the EEG channel recordings are plotted. The sampling frequency is 250Hz and the length is 6s. The time starts at -0.5s and ends at 5.5s, we therefore assume at time 0s the subject is either stimulated in some way or is signaled to do something requiring mental brain activity. As the simulations in chapter 5

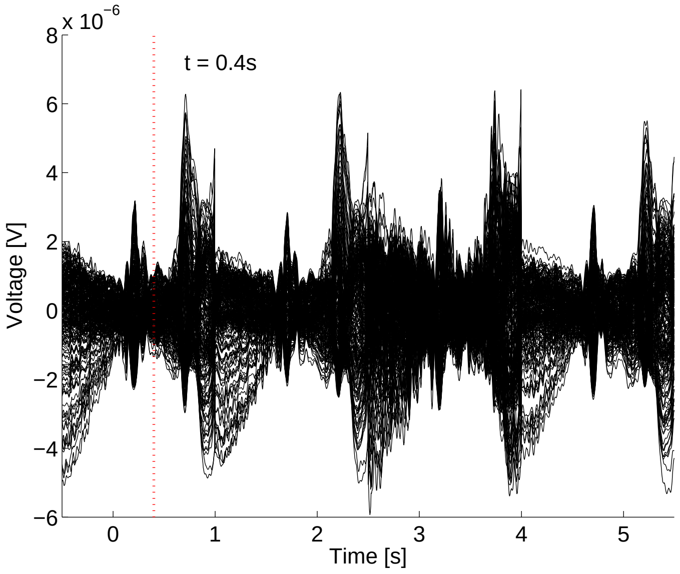

Figure D.1: *6s of 256 channel EEG recordings plotted on the same axis over a time period of 6s.*

indicated then Algorithm I requires far less number of iterations than the other algorithms. Running it for all time indices can be done in a reasonable amount of time, e.g. on 64 bit dual core AMD computer with 2GB of RAM running on Linux the run time was 20 to 30 minutes. This was done and watching the cortex activity as a movie over the whole time range showed activity on the frontal lobe, temporal lobes, motor cortex and and some minor activity at the back of the brain. Maximum number of iterations were set as 150 and the stop threshold as $\varepsilon = 10^{-3}$. Figure D.2 shows the final α and β values at each time and the number of iterations used. Note that α and β values are scaled because before running the algorithm the measurements were scaled to avoid numerical errors. This figure shows that the maximum 150 iterations are rarely used. Inspecting the inverse solutions for all time instances showed regular pulses on

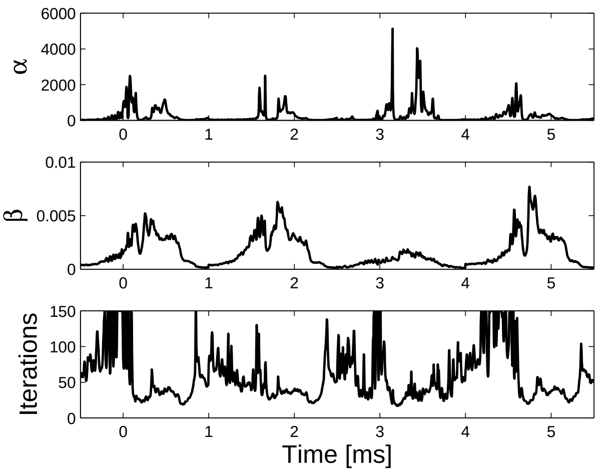

Figure D.2: *Final α and β values at each time and the number of iterations used when running Algorithm I for all time instances. α and β are scaled.*

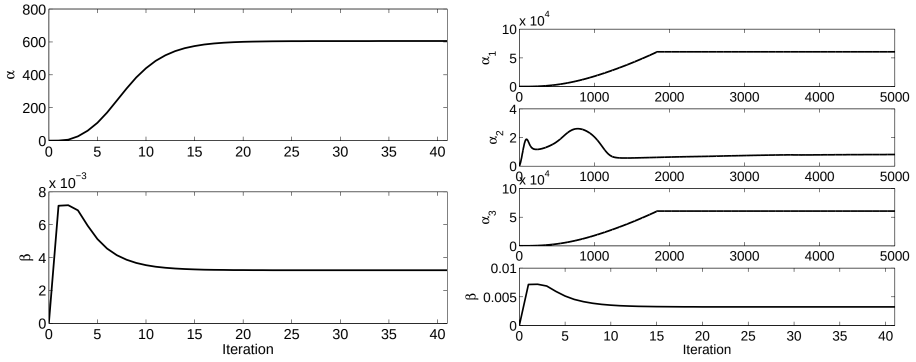

Figure D.3: *Convergence of hyperparameters for Algorithm I on the left and Algorithm IId on the right. Three α_k ($k = 1, ..., 10001$) are shown for Algorithm IId.*

the motor cortex with some underlying activity mentioned above. On figure D.3 convergence of Algorithm I and IId at time $t = 0.4s$ is plotted. Figure D.4 shows the inverse solution of Algorithm I at time 0.4s. This time index was chosen for its high overall activity. From the top view of the brain large activity is clear in the center, possibly linked with motor activity. Frontal lobe and right temporal lobe also show activity, some small activity is also visible at the back of the brain.

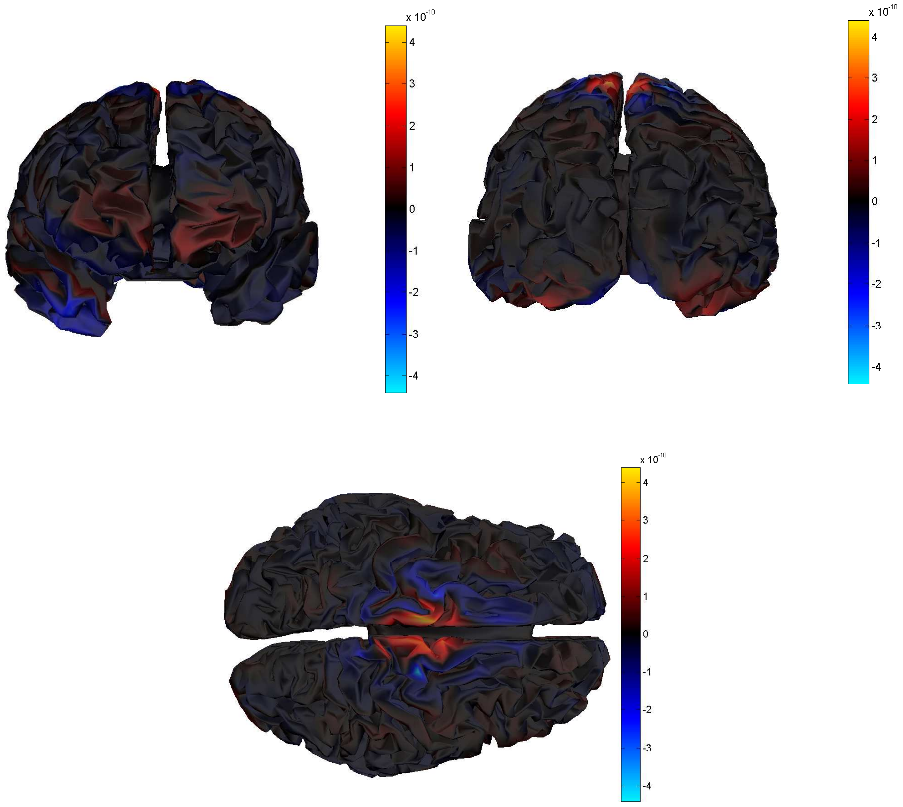

Figure D.4: *Algorithm I source estimate at time 0.4s. Top image to the left views the cortex from the front and the one to the right views it from the back. Bottom figure views it from the top.*

Algorithm IId was also run at time 0.4s, for comparison, with maximum iterations of 150 for β and 5000 for the other parameters. Stop threshold for β estimate was set as 10^{-3} and no threshold was used for the other parameters, i.e. the maximum 5000 iterations were used. The low pass filtering factor from equation 4.72 was chosen as 0.4. The algorithm converged nicely and on figure D.5 its estimate can be seen. It is a more sparse representation of the estimate from Algorithm I and shows only the strongest activities.

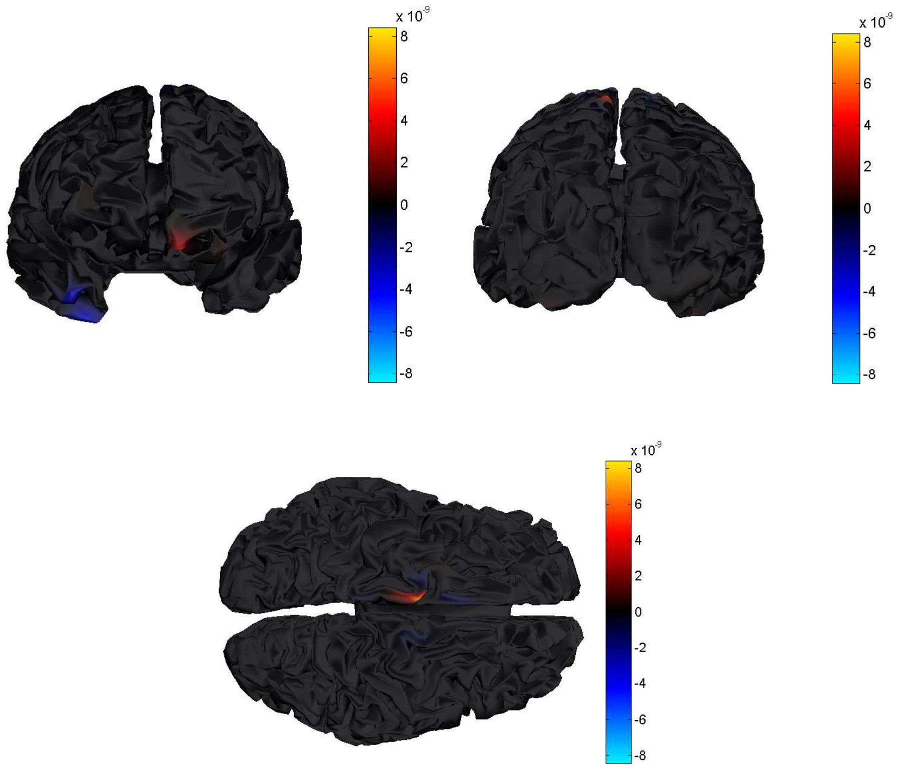

Figure D.5: *Algorithm IId source estimate at time 0.4s. Top image to the left views the cortex from the front and the one to the right views it from the back. Bottom figure views it from the top.*

D.2 Summary

Inverse calculations were done on high density EEG data supplied with the BrainStorm software package. Running Algorithm I for all time instances indicated strong pulses centered around the top of the cortical surface. These pulses could possibly be linked with motor activity in the subject. Background activity was located in the frontal part of the brain (right temporal lobe and frontal lobe) along with minor activity at the back of the brain. Algorithm IId was run on a selected time slice where similar activity as from Algorithm I was strong. The estimate from Algorithm IId complied with the one from Algorithm I showing only the strongest activities and being more sparse than the Algorithm I estimate.

Bibliography

[1] S. Baillet, J. C. Mosher and R. M. Leahy, “Electromagnetic Brain mapping,” *IEEE Signal Processing Magazine*, Nov. 2001, pp. 14-30.

[2] G. Demoment, ”Image Reconstruction and Restoration: Overview of Common Estimation Structures and Problems,” *IEEE Transaction on Acoustics, Speech, and Signal Processing*, vol. 37, no. 12, 1989.

[3] R. D. Pascual-Marqui, C. M. Michel and D. Lehmann, ”Low resolution electromagnetic tomography: a new method for localizing electrical activity in the brain,” *International Journal of Psychophysiology*, 18:49-65, 1994.

[4] R. D. Pascual-Marqui, ”Standardized low resolution brain electromagnetic tomography (sLORETA): technical details,” *Methods & Findings in Experimental & Clinical Pharmacology*, 24D:5-12 2002.

[5] M. Sato, T. Yoshioka, S. Kajihara, K. Toyama, N. Goda, K. Doya, and M. Kawato, ”Hierarchical Bayesian estiamtion for MEG inverse problem,” *NeuroImage*, vol. 23, 2004.

[6] D. Wipf, R. Ramírez, J. Palmer, S. Makeig, and B Rao, ”Analysis of Empirical Bayesian Methods for Neuroelectromagnetic Source Localization,” To appear in B. Schölkopf, J. Platt, and T. Hoffman, editors, Advances in Neural Information Processing Systems 19, MIT Press, 2007.

[7] C. M. Bishop, *Neural Networks for Pattern Recognition.* Oxford University Press. 1995.

[8] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer. 2006

[9] D. J. C. MacKay, ”Bayesian interpolation,” *Neural Computation* **4** (3), pp. 415-447, 1992.

[10] D. J. C. MacKay, ”A Practical Bayesian Framework for Backprop Networks,” *Neural Computation* **4** (3), pp. 448-472, 1992.

[11] R. M. Neal, *Bayesian Learning for Neural Networks*. Springer. 1996.

[12] M. E. Tipping, ”Sparse Bayesian Learning and the Relevance Vector Machine,” *J. of Machine Learning Research 1*, vol. 1, pp. 211-244, 2001.

[13] P. L. Nunez and R. Srinivasan, *Electric Fields of the Brain - The Neurophysics of EEG*. Oxford University Press. 2006.

[14] S. J. Williamson and L. Kaufman, “Biomagnetism,” *Journal of Magnetism and Magnetic Materials 22*, pp. 129-201, 1981.

[15] M. Hämäläinen, R. Hari, R. J. Ilmoniemi, J. Knuutila and O.V.Lounasmaa, “Magnetoencephalography - Theory, Instrumentation, and Applications ot Noninvasive Studies of the Working Human Brain,” *Reviews of Modern Physics*, vol. 65, no. 2, April 1993.

[16] M. Mørup, ”Analysis of Brain Data - Using Multi-Way Array Models on the EEG,” Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2005.

[17] R. Oostenveld and P. Praamstra, ”The five percent electrode system for high-resolution EEG and ERP measurements,” *Clin Neurophysiol*, 112:713-719, 2001.

[18] R. Oostenveld, webpage
http://oostenveld.net/
http://oase.uci.ru.nl/˜roberto/index.php/electrode/

[19] EasyCap - EEG Recording Caps and Realated Products, *http://www.easycap.de/easycap/*

[20] D. B. Geselowitz, “On Bioelectric Potentials in an Inhomogeneous Volume Conductor,” *Biophysical Journal*, vol. 7, 1967.

[21] D. B. Geselowitz, “On the Magnetic Field Generated Outside an Inhomogeneous Volume Conductor by Internal Current Sources,” *IEEE Trans. on Magnetics*, vol. mag-6, no.2, June 1970.

[22] J. Sarvas, “Basic Mathematical and Electromagnetic Concepts of the Biomagnetic Inverse Problem,” *Phys. Med. Biol.*, vol. 32, no. 1, pp. 11-22, 1987.

[23] J. C. Mosher, R. M. Leahy and P. S. Lewis, ”EEG and MEG: Forward Solutions for Inverse Mehtods,” *IEEE Trans. Biomed. Eng.,* vol. 47, pp. 1347-1355, 2000.

[24] J. J. Ermer, J. C. Mosher, S. Baillet adn R. M. Leahy, ”Rapidly recomputable EEG forward models for realistic head shapes,” *Phys. Med. Biol.,* **46**, pp. 1265-1281, 2001.

[25] S. Baillet, J. C. Mosher and R. M. Leahy, ”BrainStorm beta release: a Matlab software package for MEG signal processing and source localization and visualization,” *Neuroimage,* vol. 11, pp. S915, 2000. (http://neuroimage.usc.edu/brainstorm/)

[26] D. W. Shattuck and R. M. Leahy, ”BrainSuite: An Automated Cortical Surface Identification Tool,” *Medical Image Analysis,* 6(2):129-42, June 2002. (http://brainsuite.usc.edu/)

[27] D. L. Collins, A. P. Zijdenbos, V. Kollokian, J. G. Sled, N. J. Kabani, C. J. Holmes and A. C. Evans, ”Design and construction of a realistic digital brain phantom,” *IEEE Trans. Med. Imag.,* vol. 17, no. 3, pp. 463-468, June 1998.

[28] F. Darvas, J. J. Ermer, J. C. Mosher, R. M. Leahy, ”Generic head models for atlas-based EEG source analysis,” *Human Brain Mapping,* 27 (2): 129-143, 2006.

[29] V. S. Vladimirov, *Equations of Mathematical Physics.* Marcel Dekker, New York, USA. 1971.

[30] W. A. Strauss, *Partial Differential Equations - An Introduction.* Wiley & Sons, USA. 1992.

[31] K. B. Petersen and M. S. Pedersen, *The Matrix Cookbook.* Technical University of Denmark, February 10th 2007.

[32] C. R. Johnson, ”Positive Definite Matrices,” *American Mathematical Monthly,* vol. 77, no. 3, pp. 259-264, March 1970.

[33] R. V. V. Vidal, *Creativity and Participative Problem Solving - The Art and the Science.* Technical University of Denmark, 2006. (http://www2.imm.dtu.dk/˜vvv/CPPS/)

[34] B. Blankertz, K. R. Müller, D. Krusienski, G. Schalk, J. R. Wolpaw, A. Schlögl, G. Pfurtscheller, J. R. Millán, M. Schröder and N. Birbaumer, ”The BCI competition III: Validating alternative approachs to actual BCI problems,” *IEEE Trans. Neural Sys. Rehab. Eng.,* 14(2):153-159, 2006.

[35] B. Blankertz, G. Dornhege, M. Krauledat, G. Curio and K.-R. Müller, ”The non-invasive Berlin Brain-Computer Interface: Fast acquisition of effective performance in untrained subjects,” *NeuroImage,* 2007. (doi:10.1016/j.neuroimage.2007.01.051. to appear, available, e.g., at http://ida.first.fhg.de/bbci/#publications)

[36] L. Qin, L. Ding and B. He, ”Motor imagery classification by means of source analysis for brain-computer interface applications,” *Journal of Neural Eng.*, vol. 1, issue 3, pp. 135-141, 2004.

[37] S. K. Mitra, *Digital Signal Processing - A Computer Based Approach.* Mcgraw-Hill College. 2002