Westfälische
Wilhelms-Universität
Münster

Multilevel uncertainty quantifications for EEG source analysis

Masterarbeit

Eingereicht von:	Annegret Wittig
Datum:	27.02.2023
Erstgutachter:	Prof. Dr. Christian Engwer
Zweitgutachter:	Prof. Dr. Carsten Wolters

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

The human brain is a highly complex and fascinating organ that has been a subject of research for hundreds of years. Nowadays there exist different technologies that can be used to study the human brain. There are X-ray computed tomography (CT) and magnetic resonance imaging (MRI) which are technologies to create images of the head. Furthermore, technologies like functional magnetic resonance imaging (fMRI) or positron emission tomography (PET) analyze the metabolism inside the brain. Methods to analyze the activity of neurons are magnetoencephalography (MEG) and electroencephalography (EEG). EEG is a broadly used technique, since it provides a high temporal precision at relatively low costs. It is used in research as well as in clinical applications, e.g. as a method of presurgical epilepsy diagnosis.

EEG is a noninvasive method, it measures potential differences with sensors located at the scalp. One application of EEG is the source analysis which comprises two problems. First, there is the simulation of the EEG signal caused by a certain brain activity, the so called forward problem. On the other hand there is the reconstruction of the source of electrical potentials measured by the EEG, the so called inverse problem. Typically this includes the determination of the position and orientation of one or multiple electrical dipoles in the brain. Differential equations based on the Maxwell equations can be used to setup a mathematical model of the EEG and to formulate and solve the forward and the inverse problem.

The EEG model is deterministic but in reality it is subject to diverse uncertainties. These uncertainties have different origins. On the one hand there are uncertainties in the underlying geometry that occur since we have to use imaging methods (as the MRI) to generate a discrete image of the head that is the basis for the EEG computations. Furthermore, the conductivity of the brain tissues plays an important role in the EEG model, but its choice is uncertain due to inter- and intra-subject variability. Moreover, the potentials measured at the sensors are disturbed by noise of different origins, for example electrical potential generated in other parts of the human body as eye movement and blinks or electrical potential generated by other brain activities.

There already exist many studies trying to quantify uncertainties of the deterministic EEG model and to estimate their impact on the results of the source analysis. For example the authors of [2] and [3] study the effects of conductivity uncertainties dependent on dipole locations. Furthermore, in [13] the effects of local variations in skull and scalp thickness on the accuracy of source localization are analyzed and in [17] differences in the signal-to-noise-ratio (SNR) for varying eccentricity of the dipole are studied.

Numerous strategies to avoid and reduce these uncertainties have been developed. Typically, multiple preprocessing steps are executed before the source reconstruction is started. Preprocessing includes for example filtering out frequency bands that are not of interest, elimination of bad EEG channels and artifact removal. Nevertheless, despite all these efforts a portion of uncertainty will remain.

The EEG inverse problem is severely ill-posed, thus all source reconstruction algorithms require certain assumptions about the source model and can be classified in parametric and non-parametric solving methods. Parametric methods assume a certain number of dipoles being responsible for the current. Then they search for the best positions and orientations of these dipoles. Non-parametric methods make assumptions regarding the possible positions of dipoles. The current is represented as a linear combination of a fixed number of dipoles distributed in the whole domain with fixed position and sometimes also fixed orientation. The classical approaches of source reconstruction have in common that they return a deterministic result: the best fitting dipole(s). Considering the mentioned uncertainty studies, an assertion regarding the reliability of the reconstructed source(s) can be stated.

In this thesis a parametric but probabilistic approach is introduced which aims to reconstruct a single dipole including the uncertainty in the source analysis by reconstructing a probability distribution of the dipoles position and orientation. The main advantage of this technique is, that by including the uncertainty in the process of the source reconstruction, we have a specific quantification of uncertainty for the reconstructed source.

The probability of a dipole to be the source of a measured electric potential depends mainly on the comparison of the solution of the forward problem for this dipole with the measured values. In addition, more information can be taken into account. For example it is most likely that the dipole lies in the gray matter, the outermost layer of the brain consisting mainly of cell bodies. Furthermore a measure for the confidence in the measured data can be included. Using these information a formula of the probability distribution of the source is specified but cannot be computed since it contains the solution of the forward problem. However, evaluating the distribution at one point can be easily done. This is a typical application for Markov chain Monte Carlo methods (MCMC). These methods sample from a target distribution π by generating a Markov chain with stationary distribution π. The main idea is to generate candidates, that are accepted with a certain acceptance probability.

The most basic MCMC algorithm is the Metropolis-Hastings (MH) algorithm. There exists a variety of adaptions and extensions of this algorithm including multilevel methods. In general multilevel methods have two main advantages. On the one hand they decrease the computational costs by executing cheap operations on the low levels. On the other hand they increase the stability of the results. We will present and apply the Multilevel Delayed Acceptance (MLDA) algorithm, that has been introduced in [37] and is a combination of two other algorithms: the Multilevel MCMC (MLMCMC) algorithm [15] and the Delayed Acceptance MCMC (DA) algorithm [12].

The goal of this thesis is to apply the MH and the MLDA algorithm to reconstruct the source of a EEG-measured potential by computing the probability distribution of its position and orientation and thus quantifying its uncertainty.

In the beginning of this thesis an overview about the neurobiological background of EEG and the mathematical formulas of the EEG forward and inverse problems is given (chapter 2). In chapter 3 follows the discussion of different sources of uncertainty and an analysis of their impact on the solution of the forward and inverse problem. In the following chapter 4 the Metropolis-Hastings algorithm is explained as well as the Multilevel delayed acceptance algorithm. A major part of this thesis is the implementation of the MH and MLDA algorithms for the EEG problem using the software components DUNEuro and MUQ. In chapter 5 the theoretical and practical aspects of this implementation are described. Furthermore it contains the presentation of the results of the application of the MH and the MLDA algorithm to reconstruct unifocal sources of simulated EEG measurement values in a 2-dimensional setting. In particular the performance of the algorithms for this application has been evaluated and the computational costs of both are compared to show the benefits of a multilevel uncertainty quantification in the EEG source reconstruction. Finally, in chapter 6 the implemented methods are applied to real EEG data measured in a Somatosensory Evoked Potential (SEP) experiment.

2 EEG source reconstruction

The human brain can be seen as a highly complex network that passes information through action potentials. Electroencephalography (EEG) is a non-invasive method to measure this neural activity with sensors located at the scalp. In this section a brief introduction to the underlying neurobiological processes and the technique of the EEG is given. In this thesis the focus lies on the usage of EEG in the context of source analysis. This includes the EEG forward problem, that is the simulation of EEG measurements caused by a certain neural activity, and the inverse problem, that is the reconstruction of the source of certain EEG measurements. Both problems are formulated in this chapter, followed by a discussion of solving methods.

2.1 Neurobiological basics

There exists a lot of literature regarding the human brain and neural activity. In this section we follow mainly [62], [24] and [7]. We also refer to this literature for more background knowledge of neural processes.

The nerve system consists of two cell types: nerve cells (neurons) and glia cells. While glia cells provide support and protection, neurons process and transmit excitation. Neurons are the electrically excitable cells and thus generators of the electrical potential differences that are measured by the EEG. A neuron consists of the soma (cell body) and outgoing from this an axon (transmitter) with the axon terminal at its end and one or more branching dendrites (receivers) (see Figure 2.1). The soma regulates the cell functions and contains the cell nucleus.

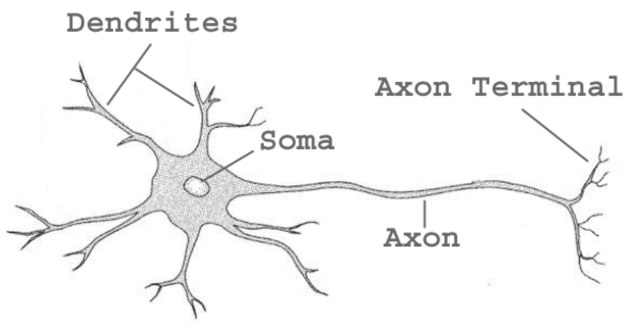


Figure 2.1: Schematic structure of a neuron taken from [1]

The neurons connect to each other via synapses and build networks, whereby synapses normally build connections between the axon and a dendrite of different cells. Neurons communicate with each other through electrochemical processes where the chemical signals use neurotransmitter substances and the electric signals use impulses which can be conducted along the axons and dendrites. They are based on potential changes at the cell membrane which occur as a result of ion transport through the permeable membrane.

Without stimulation there is a constant potential difference between the outside and the inside of the cell which is called the resting potential. This potential is a result of the ion concentration inside and outside the cell. In general, ions diffuse from areas with a high concentration to areas with a low concentration, but the cell membrane has a certain permeability to the different ions and thus allows diffusion only for specific ions. The excitation of a neuron is the disturbance of the resting potential. The permeability of the cell membrane is changed and as a result the ion transport changes which causes a rapid change in the potential at the membrane. This potential change is called the action potential. The active (presynaptic) neuron releases neurotransmitter through the synapses to the connected (postsynaptic) neurons changing locally the permeability of their membranes. That leads to a different transport of ions and results in a local current. Thus, the so called postsynaptic potential is generated. It causes transport of ions towards the soma which results, if a certain threshold is exceeded, again in a new action potential. Thus, the action potential of one neuron is delivered to another neuron. Since an axon usually has multiple synapses, multiple postsynaptic potentials are generated synchronously.

To produce a current that is large enough to be measured outside the head a high number of simultaneously active and identically oriented neurons is needed. This structure can be found in a certain group of neurons, the so called pyramidal cells. These cells have a number of basal dendrites and a large apical dendrite. This apical dendrite is oriented normally to the cortical surface. Hence, these cells are generators of the potential differences that can be measured at the head surface.

2.2 The electroencephalography

Electroencephalography (EEG) is a non-invasive method to measure brain activity by measuring potential differences on the surface of the head. Note that the acronym EEG is used as well for electroencephalogram which is the result of an electroencephalography. In this chapter we give a short overview about the basics of EEG following [48], [31] and [23].

The EEG measurement setup consists of a number of electrodes, amplifiers, an analog-to-digital (A/D) converter and a recording device [23]. The electrodes are placed directly on the scalp by using special pastes or fixed in a cap that can be easily put on (see figure 2.2). The position of the electrodes are standardized in the 10-5, 10-10 and 10-20 system. Modern EEG devices use up to 256 electrodes [27]. In general, EEG recording systems consist of several active electrodes and one reference electrode. Then the potential difference between an active electrode and the reference electrode is measured. The amplitudes of the measured potentials are of size $1 - 100\mu V$ [48]. Since these values are too small for further analysis the recorded microvolt signals are transformed by amplifiers into signals with a suitable range of voltage. Afterwards, these analog signals are converted by the A/D-converter into digital signals which are finally stored by the recording device [23].

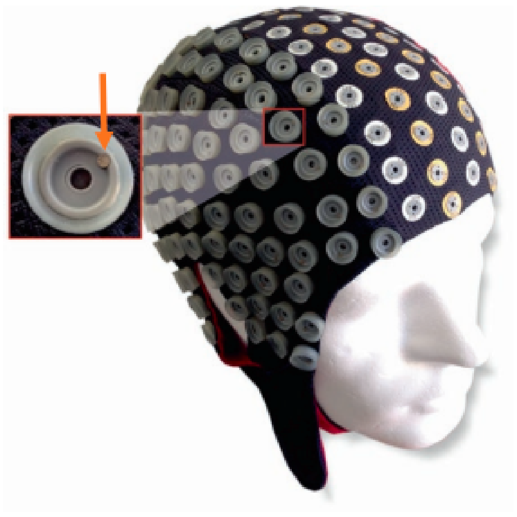


Figure 2.2: Schematic representation of an EEG cap with 256 electrodes that are arranged equidistantly. The right half shows the outside of the cap, the left side its inside. Furthermore a single electrode is enlarged. It consists of an electrode holder with a hole in the middle to fill in gel, a silicone ring that holds the gel in place and prevents the drying of the gel and an Ag/AgCl pellet electrode (orange arrow). The image is taken from [31].

We can distinguish between spontaneous and event-related EEG [31]. By spontaneous EEG we denote all brain activity that is not considered to be correlated with the internal state of the brain or with external influences like perceptions or movements. It can be seen as background activity. Event-related EEG on the other hand denotes all brain activity that is considered to have such a correlation. Figure 2.3 shows two examples of measured EEG signals, but there is a high variety of different signals in practice that are not easy to evaluate for untrained persons.

The EEG was invented in the 1920s by the German psychiatrist Hans Berger. He showed that the EEG can indeed be used for measuring brain activity outside the head [5]. Since then numerous experiments have shown that the EEG measures mainly the postsynaptic activity of cortical neurons (see chapter 2.1). In contrast action potentials

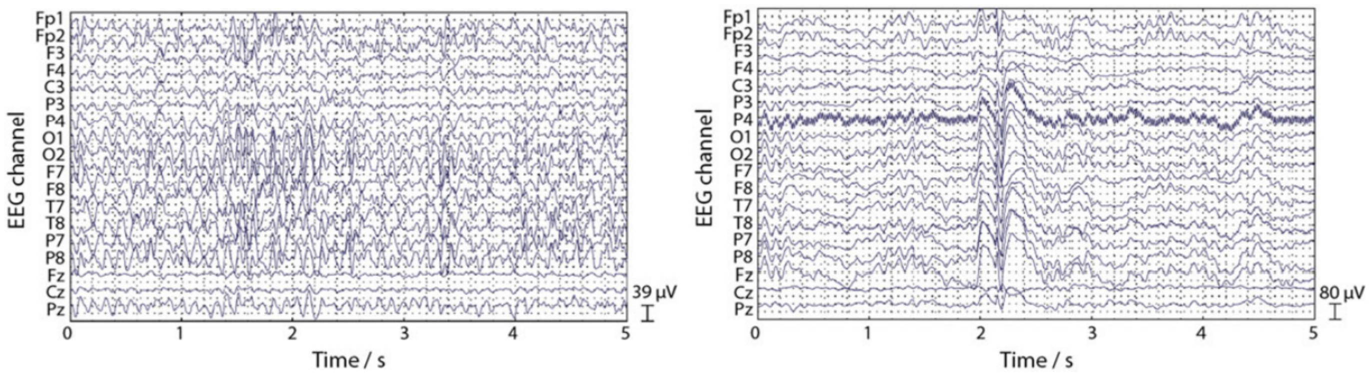


Figure 2.3: Examples of EEG signals. The left image shows background brain activity. The right image shows an interictal spike is visible which is a typical feature of a person with epilepsy. The images are taken from [10].

are not pictured by EEG nor are the activities of cortical glia cells.

Today EEG is broadly used in clinical applications as well as in the field of brain research. Typical clinical applications are the following:

- Localization and diagnosis of seizure disorders
- Confirmation of cerebral death
- Control of the depth of anesthesia
- Examination of the effectiveness of medication
- Assessment of cerebral dysfunctions after disturbed blood circulation

Benefits are especially the non-invasiveness and the high temporal resolution. Disadvantages are on the other hand the poor spatial resolution (compared to other imaging methods as computer tomography (CT), positron emitted tomography (PET) and functional magnetic resonance imaging (fMRI)) and the relative low sensitivity, which means that to reach a sufficient signal-to-noise ratio many cells need to be active [31].

2.3 Mathematical formulation

The aim of the EEG forward problem is to find the resulting sensor signals given a source in the head in dependence on a certain geometry. The corresponding inverse problem is to reconstruct neural activity in the brain given EEG measurement values. In the following we want to give a mathematical formulation of the EEG forward and the EEG inverse problem following [19], [25] and [41].

The underlying physics are formulated in the Maxwell equations:

$$\begin{aligned}
\nabla \cdot E &= \frac{\rho}{\epsilon_0} \\
\nabla \cdot B &= 0 \\
\nabla \times E &= -\frac{\partial B}{\partial t} \\
\nabla \times B &= \mu_0 (J + \epsilon_0 \frac{\partial E}{\partial t})
\end{aligned}$$

with the electric field E, the electric charge density ρ, the electric constant ϵ_0, the magnetic field B, the magnetic constant μ_0 and the current density j. Since we mostly observe frequencies below 100Hz in neural activities, we can neglect $\frac{\partial E}{\partial t}$ and $\frac{\partial B}{\partial t}$ and consider the quasi-static case of the Maxwell equations (see [19]):

$$\begin{aligned} \nabla \cdot E &= \frac{\rho}{\epsilon_0} \\ \nabla \cdot B &= 0 \\ \nabla \times E &= 0 \\ \nabla \times B &= \mu_0 J. \end{aligned} \tag{2.1}$$

The electric field can be represented as a gradient field $E = -\nabla u$ with electric potential u.

The current density J can be divided in two components, the primary current J^p and the volume current J^v. The latter is given by $J^v = \sigma \vec{E}$ where $\sigma : \Omega \to \mathbb{R}^{3\times 3}$ denotes the conductivity tensor, which can be reduced to a scalar function if we assume isotropic head compartments. The primary current is produced directly by the parallel currents flowing inside the activated cells while the volume current represents the return current flowing outside of the cells.
We can write

$$J = J^p + \sigma \vec{E} = J^p - \sigma \nabla u. \tag{2.2}$$

Taking the divergence of equation (2.2) we get

$$\nabla \cdot J^p = \nabla \cdot J + \nabla \cdot (\sigma \nabla u).$$

Since the divergence of a curl vanishes we get from equation (2.1) that $\nabla \cdot J = 0$ and thus

$$\nabla \cdot J^p = \nabla \cdot (\sigma \nabla u).$$

We assume the head to be electrically isolated, therefore, no current is flowing out of the head. We use Neumann boundary conditions to model this condition which leads us to the formulation of the EEG forward problem and the EEG inverse problem.

Definition 2.1 (The EEG forward problem)**.** *Let Ω be the head domain and $\delta\Omega$ its boundary. Let J^p be a primary current and σ the conductivity. Then the EEG forward problem is to find an electric potential u which fulfills the following Poisson equation with Neumann boundary conditions:*

$$\begin{aligned} \nabla \cdot (\sigma \nabla u) &= \nabla \cdot J^p && \textit{in } \Omega \\ (\sigma \nabla u) \cdot \vec{n} &= 0 && \textit{in } \partial\Omega \end{aligned} \tag{2.3}$$

where $\vec{n}$ denotes the unit outer normal.

Definition 2.2 (The EEG inverse problem)**.** *Let Ω be the head domain and $\delta\Omega$ its boundary. Let u be the electric potential and σ the conductivity. Then the EEG inverse problem is to find a neural activity J^p which fulfill the following Poisson equation with Neumann boundary conditions:*

$$\begin{aligned} \nabla \cdot (\sigma \nabla u) &= \nabla \cdot J^p && \textit{in } \Omega \\ (\sigma \nabla u) \cdot \vec{n} &= 0 && \textit{in } \partial\Omega \end{aligned} \tag{2.4}$$

where $\vec{n}$ denotes the unit outer normal.

The inverse problem cannot be solved analytically. There exist different methods to solve it numerically, but in each case we have to solve the corresponding forward problem many times. Thus to get a good inverse solution it is crucial to have good and fast forward solutions, which is why we spend some time on this topic before we will shortly discuss how to solve the inverse problem in chapter 2.5.

To concretize the EEG forward and the EEG inverse problem we have to specify a volume conductor model which includes the head domain Ω, its boundary $\partial\Omega$ and the conductivity $\sigma : \Omega \to \mathbb{R}$. The simplest case is the sphere-model, where the compartments are represented by concentric nested spheres. More realistic head models can be reached by using magnetic resonance imaging (MRI). In this work we assume a 5-compartment head model (see figure 2.4) with constant conductivity in each of them.

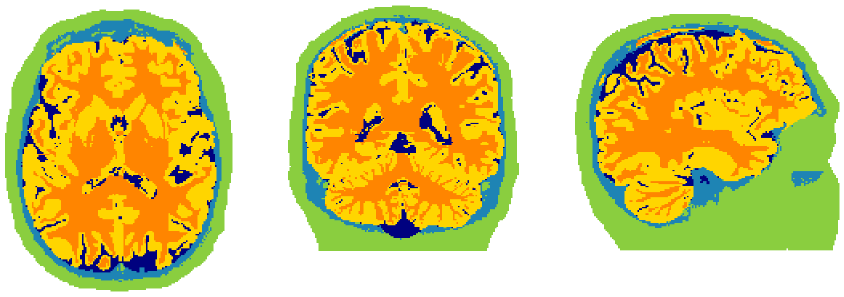


Figure 2.4: Axial (left), coronal (middle) and sagittal (right) view of a head segmented in the five compartments white matter (orange), gray matter (yellow), cerebrospinal fluid (CSF) (dark blue), skull (light blue) and scalp (green).

Furthermore, we have to find a mathematical formulation of the primary current J^p. The simplest case to model the primary current is to assume that the activated part of the brain is so small that we can represent it as a point [25]. Then we can model J^p using the Dirac delta distribution which fulfills the following properties:

$$\delta(x) = \begin{cases} \infty & \text{if } x = 0 \\ 0 & \text{else} \end{cases}$$

and

$$\int_{-\infty}^{\infty} \delta(x) = 1.$$

We set

$$J^p = M\delta(x - x_0).$$

We call x_0 the dipole position and M is the dipole moment.

2.4 Solving the forward problem

To solve the forward problem in the classical sense, strong requirements are demanded that are not met in a realistic EEG setup. In particular, continuity of the conductivity σ is required, but since we modeled it constant for each compartment, it jumps and is not continuous. To handle these conductivity jumps we need a so called weak formulation of the differential equation in (2.4). We will also discuss existence and uniqueness of a solution. Afterwards, we will introduce a discretization method to solve the weak formulation of the EEG forward problem: the finite element method (FEM).

2.4.1 Weak formulation

To obtain the weak formulation of the EEG forward problem we start with equation (2.4). We multiply both sides of the equation with a test function v and compute the integral over Ω:

$$\int_\Omega \nabla \cdot (\sigma \nabla u) v dx = \int_\Omega \nabla \cdot J^p v dx.$$

Applying partial integration gives us

$$\int_{\partial\Omega} (\sigma \nabla u) \cdot \vec{n} v dS - \int_\Omega (\sigma \nabla u) \nabla v dx = \int_\Omega \nabla \cdot J^p v dx,$$

which simplifies to

$$-\int_\Omega (\sigma \nabla u) \nabla v dx = \int_\Omega \nabla \cdot J^p v dx \tag{2.5}$$

considering the Neumann boundary condition.

Since $\int_\Omega \nabla \cdot J^p v dx$ is not well defined for $J^p = M\delta(x - x_0)$, we replace $\nabla \cdot J^p$ in the definition of the weak solution by a function $f \in L^2(\Omega)$. There are different approaches how to choose this function, which we discuss in chapter 2.4.3. At this point we just assume we will use f without specifying how it is chosen.

Since the solution should not depend on the choice of the test function v, we need a sensible choice of V which is usually the Sobolev Space $H^1(\Omega)$. The solution of equation (2.5) is only determined up to a constant. To guarantee uniqueness we add the requirement that the mean of the potential is zero over the whole domain:

$$\int_\Omega u dx = 0.$$

Hence, we use the Sobolev space with zero mean defined as

$$H^1_*(\Omega) := \left\{ u \in H^1(\Omega) : \int_\Omega u dx = 0 \right\}.$$

Putting these thoughts together and using the L_2-Norm we give a formal definition of the weak formulation.

Definition 2.3 (Weak solution of the EEG forward problem)**.** *A function $u \in H^1_*(\Omega)$ is called a weak solution of the EEG forward problem if*

$$\langle \sigma \nabla u, \nabla v \rangle_{L_2(\Omega)} = \langle f, v \rangle_{L_2(\Omega)}$$

is satisfied for all test functions $v \in H^1_*(\Omega)$.

In order to proof the existence of a solution we will use the Lax-Milgram theorem.

Theorem 2.1 (Lax-Milgram theorem [32])**.** *Let $(V, \langle \cdot, \cdot \rangle)$ be a Hilbert space. Let a be a continuous, coercive bilinear form and l a continuous linear functional. Then there exists a unique $u \in V$ such that*

$$a(u, v) = l(v) \quad \forall v \in V$$

Proof
Can be found in [32].

To apply the Lax-Milgram theorem to the weak formulation of the EEG forward problem we set $V = H^1_*(\Omega)$,

$$a(u, v) := \langle \sigma \nabla u, \nabla v \rangle_{L_2(\Omega)}$$

and

$$l(v) := \langle f, v \rangle_{L_2(\Omega)}.$$

It can be proven that our choices fulfill the requirements of the Lax-Milgram theorem (see e.g. [9]) and thus there exists a unique weak solution of the EEG forward problem as defined in definition 2.3.

2.4.2 The finite elements method

The finite elements method (FEM) is a widely used method to solve the weak formulation of a partial differential equation on a given geometry numerical. While there exist different variations of FEM the underlying ideas are similar and we want to present them in the following. The basic idea of FEM is to replace the infinite dimensional space V by a N-dimensional space V_h. Instead of a weak solution $u \in V$ we are searching a weak solution $u_h \in V_h$.

Let $\varphi_0, ..., \varphi_{N-1}$ be a basis of V_h, then we can write

$$u_h = \sum_{i=0}^{N-1} x_i \varphi_i \tag{2.6}$$

with coefficients $x_i \in \mathbb{R}$. A function $v_h \in V_h$ can be written as $v_h = \sum_{j=0}^{N-1} y_i \varphi_j$, and thus we get

$$\begin{aligned}
0 &= a(u_h, v_h) - l(v_h) \\
&= a(\sum_{i=0}^{N-1} x_i \varphi_i, \sum_{j=0}^{N-1} y_j \varphi_j) - l(\sum_{j=0}^{N-1} y_j \varphi_j) \\
&= \sum_{j=0}^{N-1} y_j \left(\sum_{i=0}^{N-1} x_i a(\varphi_i, \varphi_j) \right) - \sum_{j=0}^{N-1} y_j l(\varphi_j) \\
&= \sum_{j=0}^{N-1} y_j \left(\sum_{i=0}^{N-1} x_i a(\varphi_i, \varphi_j) - l(\varphi_j) \right).
\end{aligned}$$

Since $\varphi_0, ..., \varphi_{N-1}$ is a basis of V_h this is equivalent to

$$\sum_{i=0}^{N-1} x_i \; a(\varphi_i, \varphi_j) = l(\varphi_j) \quad \forall j = 0, ..., N-1.$$

In matrix form we write: Find a vector $x \in \mathbb{R}^N$ such that

$$Ax = l$$

with $A_{ij} = a(\varphi_i, \varphi_j)$ and $l_j = l(\varphi_j)$.

We can solve this linear system by the known methods and get a solution u_h as defined in Equation (2.6).

There are different possibilities for the choice of V_h. Usually the geometry is replaced by a mesh of for example hexahedrons or tetrahedrons. On each of these elements one (or more) functions $\varphi_0, ..., \varphi_{N-1}$ are defined. Their linear combinations build the N-dimensional vector space V_h.

2.4.3 Source models

As mentioned before a main difficulty of the EEG forward problem is modeling the mathematical dipole. The first idea was to model it using the Dirac distribution as $J^p = M\delta(x - x_0)$, but the integral $\int_\Omega \nabla \cdot J^p v dx$ that we have to compute to apply the FE method is not well defined for this choice because of the singularity. In this chapter we want to discuss three different ways of modeling the source term: the subtraction approach, partial integration and the St. Venant approach. Each of them finds another way to handle the singularity caused by the mathematical point dipole. We will mainly follow [34] and [61].

The main idea of the subtraction approach is to assume a homogenous volume conductor Ω^∞ with constant conductivity σ^∞ around the singularity and split the total potential in the singularity potential u^∞ and the correction potential u^{corr} such that

$$u = u^\infty + u^{corr}.$$

The singularity potential can be computed analytically in Ω^∞ with conductivity σ^∞ (see [61]). Subtracting the solution u^∞ from the whole Poisson problem results in the new Poisson problem

$$\begin{aligned}
\nabla \cdot (\sigma \nabla u^{corr}) &= \nabla \cdot ((\sigma^\infty - \sigma) \nabla u^\infty) && \text{in } \Omega \\
(\sigma \nabla u^{corr}) \cdot \vec{n} &= -(\sigma \nabla u^\infty) \cdot \vec{n} && \text{in } \partial\Omega.
\end{aligned}$$

Since by construction $\sigma^\infty = \sigma$ in Ω^∞ which contains the singularity, the right hand side vanishes at the singularity. Thus, the correction potential can be computed numerically using the finite element method. Consequently, using the subtraction approach, the right hand side vector of the Poisson equation is containing non-zero values on all nodes in the domain with a conductivity differing from σ^∞.

Besides the subtraction approach, there exist the so called direct approaches. One of them is partial integration, which multiplies both sides of the Poisson equation with a finite element basis φ_i

$$\int_\Omega (\sigma \nabla u)\nabla \varphi_i dx = \int_\Omega \nabla \cdot J^p \varphi_i dx.$$

Then partial integration can be applied on both sides to solve the integral over the head domain. The right hand side can be simplified using the fact, that the current density vanishes on the boundary of the domain. On the left hand side we can apply the homogeneous Neumann boundary condition. That leads us to the following equation:

$$\int_\Omega (\sigma \nabla u)\nabla \varphi_i dx = M \cdot \nabla \varphi_i(x_0).$$

For a detailed derivation see for example [61]. Thus, we replace $\nabla \cdot J^p$ by

$$f(x) = \begin{cases} M \cdot \nabla \varphi_i(x_0), & \text{if } x_0 \in supp(\varphi_i) \\ 0, & \text{else.} \end{cases}$$

Hence the right hand side vector in the Poisson equation contains only non-zero entries for the nodes of the element containing the dipole.

Another direct approach is the St. Venant method which was first introduced in [8] and originally known under the name blurred dipole approach. It follows the idea of the St. Venant principle, which is used in elasticity theory and says that "the specific (fine) details of load application do not influence the results (i.e. potentials) observed some distance away from the locus of load application." [8] Applying this idea to the EEG forward problem it says that the measured potential at the electrodes caused by the point dipole at the source location can be similar to the measurements caused by a monopole distribution around the source location. Thus, the idea of the St. Venant approach is to replace the dipole by a set of monopoles that are placed around the dipoles position. We first choose the monopole positions $x_1, ..., x_M$. Then we replace $\nabla \cdot J^p$ by

$$f(x) = \sum_{i=1}^{M} q_i \delta_{x_i}(x)$$

with delta distribution δ_{x_i} at x_i. The weights q_i are chosen such that the monopole distribution best represents the point dipole by using a Tikhonov regularization. This technique can be found in detail in [38]. The resulting right hand side vector of the Poisson equation has M non-zero entries, where M is the number of monopoles used for the approximation.

The direct approaches are computationally less expensive than the subtraction approach since the right hand side vector in the Poisson equation mainly contains zeros. On the other hand they are less reliable because they replace the dipole source model by less realistic models.

2.5 Solving the inverse problem

The EEG inverse problem is ill-posed, non-unique and unstable (see [17]) which makes finding a solving method a challenging task that requires in particular a priori assumptions about the source distribution. There exist numerous different approaches which can be divided in two groups that differ fundamentally: parametric and non-parametric solving methods. In this chapter we will introduce both types following mainly [17], [60] and [36].

2.5.1 Non-parametric solving methods

The main idea of non-parametric methods (also referred to as distributed current modeling, distributed source models or imaging methods) is to make assumptions regarding the possible positions of dipoles. We only allow a fixed number of dipoles distributed throughout the whole domain with fixed position and sometimes also fixed orientation to model the current. Then the current is represented as a superposition of the currents of all these dipoles. The task of non-parametric solving methods is to determine the coefficients of this linear combination.

Therefore the basis of non-parametric solving methods is a discretization of the EEG inverse problem from definition 2.2 where it is defined as a continuous problem assuming continuous measurement values and also a continuous source space. Since an EEG is usually measured at a certain number of fixed electrodes we assume only these measurement values as given and denote them by

$$b = \begin{bmatrix} u(\xi_1) \\ ... \\ u(\xi_S) \end{bmatrix}$$

where $\xi_1, ..., \xi_S \subset \mathbb{R}^3$ are the positions of the S electrodes. In non-parametric solving methods a discretized source space is used. This makes sense since on the one hand dipoles have a certain extent and on the other hand the resolution of EEG source reconstruction using noisy data is limited [60]. Usually we use a hexahedral or tetrahedral mesh of the domain Ω (similar as in the forward problem, see chapter 2.4.2). Either the nodes of this mesh or the centers of the cells are used as possible dipole positions. Let us assume that we have N of these positions. There are two options concerning the orientation of the dipoles: On the one hand it can be fixed (using the so called normal constraint which says that dipoles are oriented perpendicular to the cortical surface), on the other hand the orientation can be arbitrary and we allow for each dipole position three orientations which can be linear combined to every possible orientation in the three-dimensional space. Then we have a dipole strength vector $s \in \mathbb{R}^{\tilde{N}}$ with $\tilde{N} = N$ or $\tilde{N} = 3N$.

Since the differential equation in definition 2.2 is linear, we can find a linear operator L mapping the dipole strength vector on the electrode measurement values. Then we can write the inverse problem as a linear system

$$b = Ls \tag{2.7}$$

with $L \in \mathbb{R}^{S \times \tilde{N}}$. L is called the leadfield matrix and can be computed solving the forward problem in the following way. To get a column of L the forward problem from 2.1 is solved at the sensor positions $\xi_1, ..., \xi_S$ for one dipole at one of the fixed dipole positions given by the mesh with the according dipole orientation (either perpendicular or in one Cartesian direction) with unit strength. If the leadfield matrix is computed once for a certain model setting it can be used to evaluate different dipole strength vectors.

When we speak about non-parametric solving methods for the inverse problem, we consider the discrete problem from equation (2.7) where we try to find J given U and L. Basically there are two parametric approaches: regularization, where the ill-posed problem is replaced by a well-posed problem using apriori knowledge about the application (see e.g. [49]), and bayesian statistics, where the inverse problem is replaced by a statistical formulation (see e.g. [28]).

2.5.2 Parametric solving methods

Parametric methods (also referred to as focal current modeling, dipole fit models or equivalent current dipole methods) make assumptions concerning the number of dipoles. They use a so called focal current model, i.e. they assume a certain number D of dipoles (usually one up to three, see [60]) being responsible for the current. Then they search for the best positions and orientations of these dipoles. All methods consist basically of the following steps.

1. Choose D dipole positions (and orientations if not fixed).
2. Compute the resulting potential at the sensors for this hypothetical solution.
3. Compare this potential with the measured potential to evaluate the choice from step 1.

These steps are executed iteratively until a sufficiently good solution is found.

Step 2 is nothing else than solving the forward problem. Since we have to solve the forward problem many times to find a solution of the inverse problem - and the leadfield matrix can not be used here because we do not have a fixed number of dipole positions - another approach is introduced in [60] which uses the so called transfer matrix. The main idea of the transfer matrix approach is that we do not need the full solution of the forward problem but only the potential values at the sensors. Consequently, we can determine an operator giving us these values directly. Again we are interested in

$$b = \begin{bmatrix} u_h(\xi_1) \\ ... \\ u_h(\xi_S) \end{bmatrix}$$

where $\xi_1, ..., \xi_S \subset \mathbb{R}^3$ are the positions of the S electrodes. Solving the forward problem via FEM we get the coefficient vector $x \in \mathbb{R}^N$ as a result which we can use to compute the solution u_h as defined in equation (2.6). Let's assume the operator $R \in \mathbb{R}^{S \times N}$ maps the solution of the forward problem to the measurement values at the electrodes such that

$$b = Rx.$$

We replace the solution by $x = A^{-1}l$ and get

$$b = Rx = RA^{-1}l = Tl$$

with transfer matrix $T = RA^{-1} \in \mathbb{R}^{S \times N}$. To avoid the computation of A^{-1} this equation usually is transformed in to $AT^T = R^T$ using the fact that A is symmetric. Then, the $i-$th column t_i of T^T can be computed as the solution of a linear system of the form $At_i = r_i$ where r_i denotes the $i-$th column of the matrix R^T.

Instead of solving the forward problem and evaluating the solution at the electrodes, we can compute the transfer matrix once and then just assemble the right hand side l for each dipole position and compute directly $b = Tl$.

The parametric solving methods mainly differ in the way they choose possible solutions (step 1) and in the way they compare the potentials (step 3). One idea is the least-squares approach, which searches for the global minimum of the difference between the measured and the modeled electrode potential. Methods to find this minimum are for example gradient, downhill or standard simplex methods [17]. Another approach is beamforming (or spatial filtering), where the signals are filtered depending on their origin (see e.g. [4]). Its main advantage is that the number of dipoles must not be set apriori.

Remark
The transfer matrix can be used to compute the leadfield matrix in section 2.5.1 as well. Since usually there are up to a few hundred electrodes but the dimension of the source space is often of size 10^4 or more, we have $S << N$ and the transfer matrix approach can be faster. For more information see [60].

3 Uncertainties in EEG

The mathematical model of the EEG resulting in the forward and inverse problem depends on many parameters that have to be chosen carefully. Since it is a real world application, usually there does not exist the perfect choice of those parameters and thus uncertainty comes into the EEG model. In the following we want to observe three different sources of uncertainty. These are the uncertainty of the skull conductivity, the uncertainty of the underlying geometry and the uncertainty of the measured values at the sensors. In this chapter we want to discuss these uncertainties especially with regard to their origins and their impact on the results of source analysis.

3.1 Uncertainty about the conductivity

One parameter of the EEG model is the conductivity which has an impact on the result of the forward problem (see definition 2.1) and thus on the inverse problem as well. As described in section 2.3 we use constant conductivity in each of the compartments in our head model and an arising question is, how these values can be chosen. In literature different standard values can be found. Additionally, several studies show that these values can differ between persons as well (see for example [3] , [58]).

Antonakakis et al. study the inter-subject variability of skull-conductivity and skull-thickness as well as their correlation in [2]. They use a certain skull conductivity calibration procedure to determine the skull conductivity individually. In their experiments the skull-conductivity of the participants was 8.44 ± 4.84 mS/m and the skull-thickness was 5.97 ± 1.19 mm.

In [58] a table is provided listing minimal and maximal values for the conductivities of the different layers that can be found in literature (see table 3.1). It clearly shows the high variety of conductivity values.

Tissue	**Min**	**Max**	**Standard**
Skin	280.0	870.0	430.0
Skull	1.6	33.0	10.0
CSF	1769.6	1810.4	1790.0
Gray matter	220.0	670.0	330.0
White matter	90.0	290.0	140.0

Table 3.1: Values of the conductivities for the different tissues in mS/m found in literature, table taken from [58]

Thus, the choice of the exact conductivity seems to be difficult and vulnerable for uncertainty.

Hereafter, some own examinations are presented to determine the impact of changes in the conductivity values on the results of the forward problem. We use simplified test settings (adapted from the experiments in [39]) which are the following:

- We observe the 2-dimensional case.
- We use a 4-spheres head model consisting of brain, CSF, skull and scalp. Radius and conductivity of each are listed in table 3.2.
- We observe four cases of single dipoles. Following the state-of-the-art literature, we indicate the eccentricity of the dipoles, i.e. the ratio of the radius of the source position and the interface between brain and CSF. Especially high eccentricities are relevant, because in practice sources are usually located in the gray matter close to the CSF [59].

  1. Dipole 1 with an eccentricity of 98.73% and a radial orientation.
  2. Dipole 2 with an eccentricity of 98.73% and a tangential orientation.
  3. Dipole 3 with an eccentricity of 73.42% and a radial orientation.
  4. Dipole 4 with an eccentricity of 73.42% and a tangential orientation.

  In each case the strength of the dipole is assumed to be $10\mu A$.

Compartment	Radius (cm)	Conductivity (mS/m)
Brain	7.9	330.0
CSF	8.0	1790.0
Skull	8.5	10.0
Scalp	9.0	430.0

Table 3.2: Parameters of the 4-spheres head model used for the numerical experiment. While the radius values are taken from [39], we have taken the standard conductivity values from [58] (see table 3.1). Note that the value of the brain is the one of the gray matter since we only assume a 4-spheres model.

To evaluate the results we first introduce common error measures following [59] and [41].

- Relative L2-error:
$$L2(u, u_{err}) = 100 \frac{\|u - u_{err}\|_2}{\|u\|_2} \% \tag{3.1}$$
- Relative difference measure (RDM):
$$RDM(u, u_{err}) = 50 \left\| \frac{u_{err}}{\|u_{err}\|_2} - \frac{u}{\|u\|_2} \right\|_2 \% \tag{3.2}$$
- Magnitude error (MAG):
$$MAG(u, u_{err}) = 100 \left(\frac{\|u_{err}\|_2}{\|u\|_2} - 1 \right) \% \tag{3.3}$$

As the relative L2-error is a general measure to compare the forward solutions, RDM and MAG make more specific statements about the differences of the two solutions. The RDM measures differences in the topography while the MAG indicates differences in the overall signal magnitude compared to the reference solution. For EEG source analysis that means that the RDM is related to errors in the position and the orientation of the reconstructed source and the MAG is related to errors in the magnitude of the reconstructed source. [59] states that "a higher RDM correlates with a less accurate source localization" and thus "a low RDM is of high importance for almost all applications of EEG source analysis."

In our experiment we execute the following steps.

1. Solve the forward problem for the standard conductivity values taken from table 3.1 to get a reference solution u.
2. For each compartment: Solve the forward problem to get u_{min} using the minimal conductivity value for this compartment (taken from table 3.1) while the standard conductivity values are remained for the other compartments.
3. Compute the error measures L2-error, RDM und MAG as defined above.
4. Repeat steps 2 and 3 with the maximum conductivity value (taken from table 3.1) to compute u_{max} and the corresponding error measures for each compartment.
5. For each compartment choose the maximal L2-error, RDM and MAG of the two computed values.

The results can be found in table 3.3. They show clearly that there is a high impact of the conductivity values on the solution of the forward problem. There exists no strong dependency of the error measures on the position or orientation, but they differ considerably between the compartments.

We are mainly interested in the RDM, because it is a measure of the difference in the distribution of the potential at the electrodes. Especially changes in the skull conductivity result in a high RDM.

These results correspond to the findings of [3]. The authors conclude that their studies "indicate that source localization is most sensitive to the conductivities of the skull and scalp". Considering the inverse problem, studies found

	Brain	**CSF**	**Skull**	**Scalp**
Dipole 1				
L2 error	47.3%	0.1%	47.9%	22.7%
RDM	2.3%	0.0%	19.9%	7.9%
MAG	47.2%	0.1%	35.8%	17.5%
Dipole 2				
L2 error	47.6%	0.1%	43.1%	19.0%
RDM	1.6%	0.0%	15.4%	5.7%
MAG	47.6%	0.1%	35.2 %	15.8%
Dipole 3				
L2 error	46.4%	0.1%	53.8%	25.5%
RDM	1.7%	0.0%	15.3%	6.5%
MAG	46.3%	0.1%	49.2%	22.8%
Dipole 4				
L2 error	47.0%	0.1%	46.8%	20.6%
RDM	1.0%	0.0%	10.6%	4.5%
MAG	47.0%	0.1%	44.1%	19.0%

Table 3.3: Results of the experiment to estimate the impact of changes in the conductivity for dipoles of different orientations (dipoles 1 and 3 are radial, dipoles 2 and 4 are tangential) and different eccentricities (dipoles 1 and 2 have an eccentricity of 98.73%, dipoles 3 and 4 have an eccentricity of 73.42%).

similar results: "Small changes on skull conductivity can cause substantial attenuations on the modeled electric fields resulting in localization errors in the centimeter range and orientation changes of more than 25°."[2]

As a consequence of these results multiple studies state that the skull conductivity of the particular person has to be determined before a source analysis can be done (see for example [14] and [2]). The authors of [2] have developed a procedure to estimate the skull conductivity of a patient. This procedure uses a combination of EEG/MEG and MRT and is non-invasive and fast. Since there already exists this method to eliminate this source of uncertainty we will not observe it further in this thesis.

3.2 Uncertainty regarding the geometry

Solving the forward problem and thus solving the inverse problem as well, is strongly dependent on the underlying head geometry, because the conductivity at each point is given by the allocated compartment at this point. In general, the geometry is constructed from an MRI image. Within such an image each voxel is assigned to a gray value. Then by considering these gray values and applying additional methods as pattern recognition the tissues are identified. As a result of this process a tissue probability map is generated which gives the probability that a voxel belongs to a tissue and thus contains a probability per voxel per tissue. Using this tissue probability map a final head model can be constructed.

While the other tissue boundaries can be recognized well from MRI images, it is especially hard to recognize the inner skull boundary from such an image [22]. In addition, the conductivity of the skull is much smaller than the conductivity of the other tissues (see chapter 3.1). Thus, it acts as an isolating layer decreasing and blurring the potential towards the electrodes. As a result, the geometry of the skull (especially its inner boundary) is the most critical part for errors in the geometry [22].

A numerical experiment where we assume the geometry of the skull to be uncertain, should help to estimate the impact of these geometry uncertainties. We use the same 2-dimensional 4-spheres head model as in the previous section (section 3.1). Since there is a direct relation between the skull thickness and the skull conductivity we model this assumption by choosing an uncertain skull conductivity. We model the error as a normal distributed multiplicative error

$$\sigma_{skull}^{err} = \epsilon \cdot \sigma_{skull}$$

with $\epsilon \sim \mathcal{N}(1, \rho^2)$.

First, we model the conductivity error for each cell of the mesh independent and thus determine a new erroneous conductivity for each cell. We execute the following steps:

1. Compute reference solution u_{ref} by solving the forward problem with undisturbed skull conductivity.
2. For each cell draw $\epsilon \sim \mathcal{N}(1, \rho^2)$ and set $\sigma_{skull}^{err} = \epsilon \cdot \sigma_{skull}$ in this cell.
3. Solve forward problem to get the potential u.
4. Compute error measures: L2-error (see equation (3.1)), RDM (see equation (3.2)) and MAG (see equation (3.3)).
5. Run steps 2 to 4 n times.

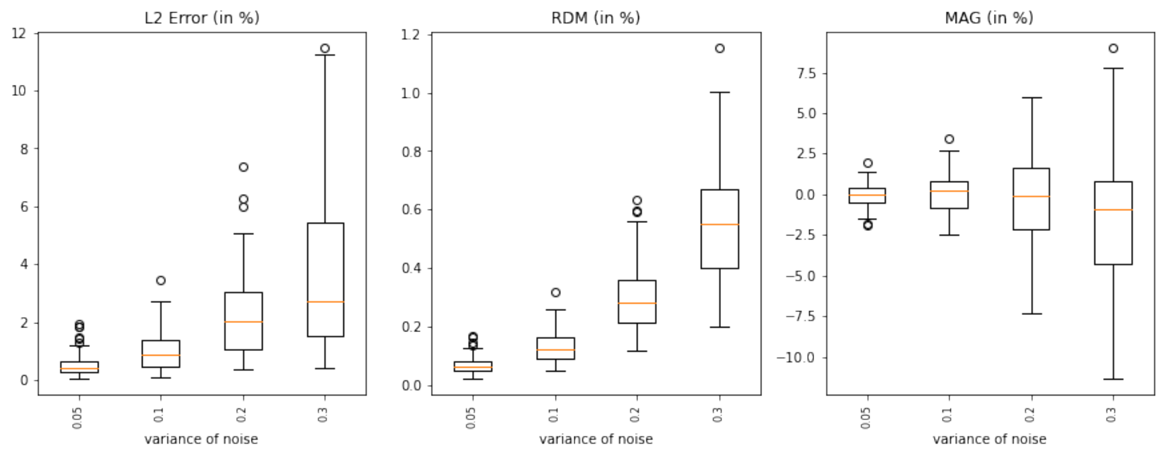
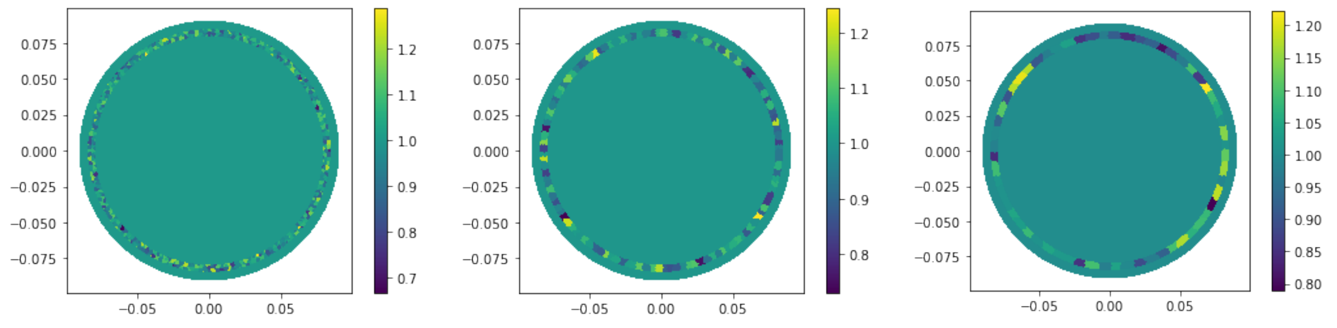


Figure 3.1: L2-error, RDM and MAG of the potential u computed assuming an erroneous skull conductivity with a normal distributed error $\epsilon \sim \mathcal{N}(1, \rho^2)$ dependent on the variance ρ^2 for $n = 100$ trials.

The results of this experiment can be seen in figure 3.1. As might be expected the error measures increase for increasing error variance. We observe a relevant L2-error, but analyzing the RDM and MAG shows that it is mainly caused by a difference in the magnitude and that we have a low RDM even for a high error variance. This indicates a low impact on the position and orientation of the source in the inverse problem.



(a) No local dependency of the error. (b) Low local dependency of the error. (c) High local dependency of the error.

Figure 3.2: Visualization of the multiplicative error $\epsilon \sim \mathcal{N}(1, 0.1)$ of the conductivity. It varies in the skull, in the other compartments conductivity is not assumed to be random but deterministic and thus the error is set to 1.

Now we assume a certain spatial dependency of the errors which seems more realistic since we want to model geometry errors as a varying skull thickness by areas of lower and higher conductivity. To implement this assumption we combine cells of the skull at the same (or a very similar) angle to one area with a constant error. We start with a low local dependency (120 areas of approximately the same size) and go on with a high local dependency (only 60 areas are considered) as illustrated in figure 3.2.

In figure 3.3 boxplots of the RDM are plotted for the three cases of spatial dependency (no dependency, low dependency, high dependency) and for different variances of the multiplicative error. These results show that as expected the RDM increases when we apply a spatial dependency on the noise. Though, it is still not much above 1% for a very strong spatial dependency. Thus, we can not see a high impact on the results of the EEG forward problem and consequently the impact on the position and orientation of a reconstructed source is insignificant as well.

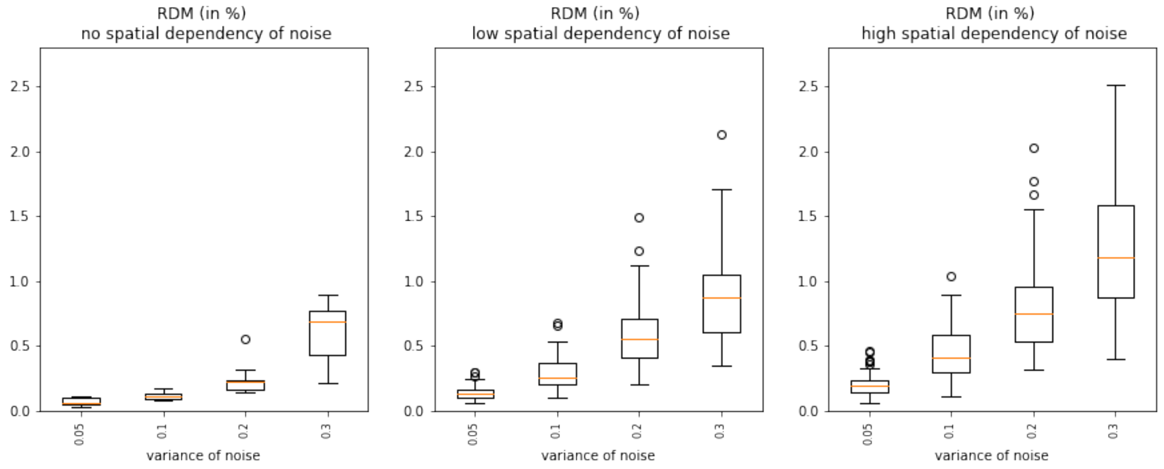


Figure 3.3: RDM of the potential u computed assuming an erroneous skull conductivity with different strong spatial dependencies of the multiplicative error. The RDM is visualized in dependency of the variance of the normal distributed error. The boxplots show, that even for a strong spatial dependency the RDM is growing slow.

Similar results are reached in [16] where the authors conclude that their "results show that the effect of the stochastic head model on the estimation of the source position parameters is an uncertainty with standard deviation of the same order as the standard deviation of the perturbations themselves. When compared with the effect of noise in the measurements, the perturbations on the shape of the head have second-order effect on the EEG inverse problem." Furthermore [13] concludes that their results "indicate that local variations in skull and scalp thickness cause EEG localization errors which are generally much less than 1 cm."

The above analysis of the uncertainty caused by geometry uncertainties leads us to the result, that this type of uncertainty is not observed further in this thesis since it has no serious impact on the reconstruction of the source.

3.3 Uncertainties in measurement values

Many types of noise are present in the data measured by EEG [63]. They can be grouped by their origin in external and internal noise. External noise means all noise that comes from outside the brain. This includes electrical potential generated in other parts of the human body as for example eye movement and blinks or heart activity, that generate electrical potential that is higher than the electrical potential generated by the brain which is very small (in the size of micro volt). A detailed discussion of those artifacts can be found in [26]. Furthermore, there can be noise from the experimental environment, but usually this can be removed by an appropriate experimental setup.

Internal noise on the other hand is produced inside the brain. It is caused by the fact that the brain is executing many different tasks at the same time and each of those is producing electrical activity. If we are interested in analyzing one particular signal this is overlaid by noise produced by other activities.

A common measure to quantify noise is the signal-to-noise-ratio (SNR) which measures the ratio of meaningful signals and meaningless (or unwanted) signals. It can be determined from EEG measurements and differs for each experiment since it depends on the experimental setup as well as on the experimental subject. In theoretical studies a SNR has to be set - for example in [54] a SNR of $3-43$ dB is set and in [45] a SNR of $20-30$ dB is set.

There exist many different strategies to handle noise in the measurement values. In [44] four possibilities are listed:

- The elimination of noise sources
- Averaging over multiple trials
- The rejection of noisy data
- The removal of noise from the data

These strategies help to reduce noise but it won't be possible to remove noise completely. For this reason we want to include this uncertainty in the mathematical formulation of our problem.

The task of the deterministic inverse problem using a non-parametric solving method as described in chapter 2.5.1 is, given the sensor values $b \in \mathbb{R}^m$, to find a source configuration $s \in \mathbb{R}^n$ such that

$$b = Ls,$$

where $L \in \mathbb{R}^{m \times n}$ is the leadfield matrix (see equation (2.7)).

We now assume that the measurement values at the sensors are insecure. We model this insecurity following [36] as a normal distributed additive noise and rewrite the inverse problem as

$$B = LS + \mathcal{E}$$

with $\mathcal{E} \sim \mathcal{N}(0, \Gamma)$ for covariance matrix $\Gamma = diag(\gamma_1^2, ..., \gamma_m^2) \in \mathbb{R}^{m \times m}$ where γ_i^2 is the noise variance at sensor i. Note that B and S are random variables now, too.

The probability of measuring the values b given the source configuration s is then given by

$$p(b|s) = \frac{1}{\sqrt{(2\pi)^m |\Gamma|}} \exp\left(-\frac{1}{2}(b - Ls)^T \Gamma^{-1} (b - Ls)\right)$$

where $|\Gamma|$ denotes the determinant of the covariance matrix Γ. If the noise variance is equal for all sensors, i.e. $\gamma_1^2 = ... = \gamma_m^2 = \gamma^2$ this equation simplifies to

$$p(b|s) = \left(\frac{1}{2\pi\gamma^2}\right)^{\frac{m}{2}} \exp\left(-\frac{1}{2\gamma^2} \|b - Ls\|_2^2\right).$$

We call this probability the likelihood distribution and denote it by $p_{li}(b|s)$.

The probability of having a source configuration s given some measurement values b can not be computed directly since $m << n$. We could use Bayes rule to compute it by

$$p_{post}(s|b) = \frac{p_{li}(b|s) p_{pr}(s)}{p(b)}$$

where $p_{pr}(s)$ is the so called prior distribution. For this approach it is necessary to choose an appropriate prior distribution $p_{pr}(s)$ and the distribution $p(b)$ that gives the probability for measuring specific values.

We will use a different approach to generate samples of the posterior-distribution: the Markov Chain Monte Carlo method. This method will be presented in the next chapter.

4 Markov chain Monte Carlo methods

Suppose we want to sample from a certain target distribution π. It is a common case that sampling from this distribution is impossible (or very complicated) while evaluating the distribution at one point can be easily done. This is a problem that can be solved using Markov chain Monte Carlo methods (MCMC). These methods sample from the target distribution by generating a Markov chain with stationary distribution π.

In this section we explain the elegant idea of MCMC in detail and present the Metropolis-Hastings (MH) algorithm on which the other MCMC algorithms are based. We will go on with the introduction of different extensions of the MH algorithm leading to the Multilevel delayed acceptance algorithm (MLDA). Furthermore we discuss different diagnostic and evaluation methods for MCMC algorithms. But since it is the basis of all the ideas we start with a brief introduction to Markov chains.

4.1 Markov chains

The following definitions concerning Markov chains are taken from [29], [30], [46] and [6] to which we also refer for more background information. Since Markov chains are special types of discrete-time stochastic processes we start with a formal definition of those.

Definition 4.1 (Discrete-time stochastic process)**.** *Let $(E, \mathcal{E})$ be a measurable space. We call a sequence $\{X_n\}_{n\in\mathbb{N}}$ of random variables with values in $(E, \mathcal{E})$ a discrete-time stochastic process with state space E.*

In the following we assume $E \subset \mathbb{R}^d$ with Borel $\sigma-$algebra $\mathcal{E} = \mathcal{B}(E)$. The characterizing property of Markov chains is that all future states only depend on the present state but not on the past states. Since it is the essential property of Markov chains it is called the Markov property.

Definition 4.2 (Markov property)**.** *A discrete-time stochastic process $\{X_n\}_{n\in\mathbb{N}}$ with state space E fulfills the Markov property if*

$$\mathbb{P}[X_{n+1} \in A | X_1, ..., X_n] = \mathbb{P}[X_{n+1} \in A | X_n] \quad \forall A \in \mathcal{B}(E)\ \forall n \in \mathbb{N}.$$

Definition 4.3 (Homogeneity)**.** *A discrete-time stochastic process $\{X_n\}_{n\in\mathbb{N}}$ with state space E fulfilling the Markov property is called homogeneous if the transition probability is independent of the time*

$$\mathbb{P}[X_{n+1} \in A | X_n] = \mathbb{P}[X_{m+1} \in A | X_m] \quad \forall A \in \mathcal{B}(E)\ \forall n, m \in \mathbb{N}.$$

A homogeneous Markov chain is fully defined by its initial distribution and the transition probabilities. The initial distribution is given by a probability measure $\mu_0(\cdot)$ such that

$$\mathbb{P}[X_0 \in A] = \mu_0(A)$$

for all $A \in \mathcal{B}(E)$. The transition probabilities are given by a transition probability kernel.

Definition 4.4 (Transition probability kernel)**.** *A function $P : E \times \mathcal{B}(E) \to [0, 1]$ is called a transition probability kernel, if the following two conditions are fulfilled.*

(i) $P(\cdot, A)$ is a measurable function on E for all $A \in \mathcal{B}(E)$.

(ii) $P(x, \cdot)$ is a probability measure on $\mathcal{B}(E)$ for all $x \in E$.

The transition probability kernel $\mathcal{P}(x, A)$ gives the probability of the process to jump directly from state x to a state contained in the set A, hence we set

$$\mathbb{P}[X_{n+1} \in A | X_n = x] := P(x, A) \quad \forall A \in \mathcal{B}(E)\ \forall n \in \mathbb{N}.$$

Note that the transition probability does not depend on n which is the homogeneity property.

Notation
Since $P(x, \cdot)$ is a probability measure on $\mathcal{B}(E)$ for all $x \in E$ we write

$$P(x, A) = \int_A P(x, dy).$$

Definition 4.5 (Composition of transition probability kernels). *Let P_1 and P_2 be transition probability kernels. We define their composition as*

$$(P_1 \circ P_2)(x, A) = \int_E P_1(y, A) P_2(x, dy) \quad \forall A \in \mathcal{B}(E), x \in E.$$

Now we have

$$\mu_1(A) = \mathbb{P}[X_1 \in A] = \int_E P(y_0, A)\mu_0(dy_0)$$

and recursively

$$\mu_n(A) = \mathbb{P}[X_n \in A] = \int_E P(y_{n-1}, A)\mu_{n-1}(dy_{n-1}).$$

These thoughts are put together in the following definition.

Definition 4.6 (Homogeneous Markov chain). *Let $(E, \mathcal{E})$ be a measurable space and $\mathcal{P}$ a transition probability kernel defined on it. A discrete-time stochastic process $X = \{X_n\}_{n \geq 0}$ with state space E is called a homogeneous Markov chain with Markov kernel $\mathcal{P}$ and initial distribution μ_0 if*

$$\mathbb{P}[X_0 \in A_0, ..., X_n \in A_n] = \int_{A_0} ... \int_{A_{n-1}} P(y_{n-1}, A_n)P(y_{n-2}, dy_{n-1})...P(y_0, dy_1)\mu_0(dy_0).$$

Theorem 4.1 (Existence of Markov chain). *For any initial distribution μ and any transition probability kernel $\mathcal{P}$ there exists an associated Markov chain.*

Proof
The proof of this theorem can be found in [29].

In the following we will define some important properties of Markov chains we will need later. The definitions follow [46] where also illustrative examples can be found.

Definition 4.7 (Properties of Markov chains). *Let $X = \{X_n\}_{n \in \mathbb{N}}$ be a Markov chain with state space E.*

(i) X is called stationary with stationary distribution $\mu(\cdot)$ if

$$\mu(A) = \int_E P(x, A)\mu(dx) \quad \forall A \in \mathcal{B}(E).$$

(ii) X is called reversible with respect to $\mu(\cdot)$ if

$$\int_B P(x, A)\mu(dx) = \int_A P(x, B)\mu(dx) \quad \forall A, B \in \mathcal{B}(E).$$

(iii) X is called ϕ-irreducible if there exists a non-zero $\sigma-$finite measure ϕ on E such that for all $A \subset E$ with $\phi(A) > 0$ and for all $x \in E$ there exists $n(x, A) \in \mathbb{Z}^+$ such that $P^n(x, A) > 0$.

(iv) Let X be stationary with stationary distribution μ. Then X is called d-periodic if there exists a $d \geq 2$ and disjoint subsets $E_1, ..., E_d \subset E$ with

$$\begin{aligned} P(x, E_{i+1}) &= 1 \quad \forall x \in E_i \ \forall i = 1, ..., d-1, \\ P(x, E_1) &= 1 \quad \forall \in E_d \end{aligned}$$

such that $\mu(X_1) > 0$. If there exists no such d X is called aperiodic.

Remark

1. It is common to say X is in detailed balance with $\mu(\cdot)$ if X is reversible with respect to $\mu(\cdot)$. Both designations are used in literature and are identical in their meaning.

2. Since $P(x, A) = \int_A P(x, dy)$ X is reversible with respect to $\mu(\cdot)$ iff

$$\mu(dx)P(x, dy) = \mu(dy)P(y, dx) \forall x, y \in E.$$

Theorem 4.2. *If a Markov chain $X = \{X_n\}_{n \geq 0}$ is reversible with respect to a probability distribution $\mu(\cdot)$ then X is stationary with stationary distribution $\mu(\cdot)$.*

Proof
Since $P(x, E) = 1$ it yields

$$\int_E P(x, A)\mu(dx) = \int_A P(x, E)\mu(dx) = \int_A \mu(dx) = \mu(A).$$

□

Theorem 4.3. *Let X_1 and X_2 be two Markov chains with kernels P_1 and P_2 that both are reversible with respect to μ and commute. Then the Markov chain with their composition $P_1 \circ P_2$ as kernel is also in detailed balance with μ.*

Proof
Since they are reversible with respect to μ the kernels P_1 and P_2 satisfy

$$\mu(dx)P_1(x, dy) = \mu(dy)P_1(y, dx)$$

and

$$\mu(dx)P_2(x, dy) = \mu(dy)P_2(y, dx).$$

Using these equations and the commutativity we get

$$\begin{aligned}
\mu(dx)(P_1 \circ P_2)(x, dy) &= \mu(dx) \int_E P_1(z, dy)P_2(x, dz) \\
&= \mu(dx) \int_E P_2(x, dz)P_1(z, dy) \\
&= \mu(dx) \int_E P_2(z, dx)\frac{\mu(dz)}{\mu(dx)}P_1(y, dz)\frac{\mu(dy)}{\mu(dz)} \\
&= \int_E P_2(z, dx)P_1(y, dz)\mu(dy) \\
&= \mu(dy)(P_1 \circ P_2)(y, dx).
\end{aligned}$$

Thus $P_1 \circ P_2$ is reversible with respect to μ.

□

Definition 4.8 (n-step transition probability)**.** *The n-step transition probability is defined as the probability of the process to jump from state x to a state in the set A in n steps and is denoted by*

$$P^n(x, A) = \mathbb{P}[X_n \in A | X_0 = x] \quad \forall A \in E \; \forall n \in \mathbb{N}.$$

Theorem 4.4. *The n-step transition probability is given recursively by*

$$P^0(x, A) = \delta_x(A) = \begin{cases} 0, & \text{if } x \notin A \\ 1, & \text{if } x \in A \end{cases}$$

and

$$P^n(x, A) = (P \circ P^{n-1})(x, A) = \int_E P(z, A)P^{n-1}(x, dz)$$

for all $n > 0$.

Theorem 4.5. *If a Markov chain $X = \{X_n\}_{n \geq 0}$ with values in $(E, \mathcal{B}(E))$ is ϕ-irreducible, aperiodic and stationary with stationary distribution π then*

$$\lim_{n \to \infty} P^n(x, A) = \pi(A)$$

for all measurable $A \subseteq E$.

Proof
The proof of this theorem can be found in [46].

4.2 Basics of MCMC

As its name implies MCMC combines two mathematical ideas: Markov chains and Monte Carlo Sampling. The idea behind Monte Carlo Sampling is to estimate properties of a distribution by drawing random samples from it and evaluating their properties. A simple example is, that we can estimate the expected value of a distribution by drawing a sufficiently large number of random samples from it and computing the sample mean of them. We want to use the Monte Carlo method to reconstruct a distribution by drawing samples from it and determine their empirical distribution. The ideas of this chapter follow [18].

Definition 4.9 (Empirical distribution)**.** *Let $\theta = (\theta^0, ..., \theta^n)$ be a sample of size n with observations of dimension d, i.e. $\theta^i = (\theta^i_1, ..., \theta^i_d)$. The empirical distribution function $F_n : \mathbb{R}^d \to [0,1]$ is defined by*

$$F_n((x_1, ..., x_d)) := \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{\theta^i_1 \leq x_1, ..., \theta^i_d \leq x_d\}}.$$

The Markov chain idea is that the observations θ^i used for the Monte Carlo Sampling are generated by a special sequential process which builds a Markov chain fulfilling two conditions.

1. It is stationary with stationary distribution π where π is the distribution we want to sample from.
2. It converges towards its stationary distribution.

If we have such a Markov chain and randomly walk it, we sample indeed from π. But how can we construct a Markov chain with this stationary distribution?

To generate a Markov chain fulfilling the first conditions, we can use theorem 4.2. Thus the probability transition kernel $P(x, A)$ of our Markov chain has to be reversible with respect to π to guarantee that it is stationary with stationary distribution π. Suppose we represent the probability transition kernel as

$$P(x, dy) = p(x, y)dy + r(x)\delta_x(dy) \tag{4.1}$$

for an arbitrary function p with $p(x, x) = 0$ and the function $r(x) := 1 - \int_E p(x, y)dy$ (see [18]).

Then we assume the following reversibility constraint

$$\pi(x)p(x, y) = \pi(y)p(y, x) \tag{4.2}$$

and show that it is sufficient for the stationarity of $P(x, \cdot)$ with stationary distribution π.

Theorem 4.6. *If $p(x, y)$ satisfies the reversibility constraint (equation (4.2)), then π is the stationary distribution of $P(x, \cdot)$.*

Proof The proof is taken from [18].

$$\begin{aligned}
\int_E P(x, A)\pi(dx) &= \int_E \left(\int_A P(x, dy) \right) \pi(dx) \\
&= \int_E \left(\int_A p(x, y)dy + r(x)\delta_x(A) \right) \pi(dx) \\
&= \int_E \left(\int_A p(x, y)\pi(x)dy \right) dx + \int_E r(x)\delta_x(A)\pi(x)dx \\
&= \int_E \left(\int_A p(y, x)\pi(y)dy \right) dx + \int_A r(x)\pi(x)dx \\
&= \int_A \left(\int_E p(y, x)dx \right) \pi(y)dy + \int_A r(x)\pi(x)dx \\
&= \int_A (1 - r(y))\, \pi(y)dy + \int_A r(x)\pi(x)dx \\
&= \int_A \pi(x)dx \\
&= \pi(A)
\end{aligned}$$

□

Now we can answer the question how a Markov chain with a certain stationary distribution can be constructed. We simply define the probability transition kernel as in equation (4.1) and ensure that $p(x, y)$ satisfies the reversibility constraint from equation (4.2), then we have found the desired Markov chain. This is the basis of all MCMC algorithms.

To generate a Markov chain that fulfills the second condition we use theorem 4.5 and thus have to ensure that our Markov chain is $\phi-$irreducible and aperiodic.

4.3 The Metropolis-Hastings algorithm

In the following we want to discuss the general Metropolis-Hastings algorithm which is the standard MCMC algorithm. It is described in algorithm 4.1 following [37]. Starting at an appropriate starting point we generate a new candidate using the proposal distribution. Then we compute the acceptance probability by comparing the most recent sample and the new candidate. We decide randomly if we accept or reject the proposal using the computed acceptance probability.

Algorithm 4.1 Metropolis-Hastings

```
  function
  $(\theta_1, ..., \theta_N) = \mathrm{MH}(\pi_t, q(\cdot|\cdot), \theta_0, N)$

  Input
      target density $\pi_t$,
      proposal density $q(\cdot|\cdot)$,
      starting point $\theta_0$,
      number of samples $N$

  Output
  list of samples $(\theta_1, ..., \theta_N)$

  Algorithm
     for i = 1,...,N do
        Generate a new candidate $\psi \sim q(\cdot|\theta_{i-1})$.
        Compute the acceptance probability
            $\alpha(\psi, \theta_{i-1}) = \min\left(1, \frac{\pi_t(\psi)q(\theta_{i-1}|\psi)}{\pi_t(\theta_{i-1})q(\psi|\theta_{i-1})}\right).$
        Generate an equal distributed random number $u \in [0, 1]$.

        if $u \leq \alpha$ then accept candidate $\psi$ and set $\theta_i = \psi$.
        else reject candidate $\psi$ and set $\theta_i = \theta_{i-1}$
```

Remark

(i) If the proposal density $q(\cdot|\cdot)$ is symmetric (e.g. a random walk), the acceptance probability can be written as

$$\alpha(\psi, \theta_{i-1}) = \min\left(1, \frac{\pi_t(\psi)}{\pi_t(\theta_{i-1})}\right).$$

In that case the acceptance probability is the comparison of the posterior density of the most recent sample and the new candidate. If the posterior of the candidate is higher than the posterior of the last accepted state we accept it as a new candidate. If it is lower we accept it with the probability of the ratio of both posteriors.

(ii) It is possible that the starting point lies in a region with a very low probability and thus the first few samples oversample this region, because the chain has not converged, yet. To handle this issue, it is common to extend the algorithm by defining a so called burn-in phase. All samples generated in this phase are discarded (see [20]). Actually it is not necessary to have a burn-in phase because we will show the convergence of the Markov chain. In practice it can accelerate the algorithm in the way that rejecting a number of draws can be much cheaper than generating a multiple of draws to average out the effects of those first samples.

Theorem 4.7. *The MH algorithm described in algorithm 4.1 generates a Markov chain $\theta = (\theta_1, ..., \theta_N)$ with stationary distribution $\pi_t(\cdot)$.*

Proof
First note that the MH algorithm generates in each iteration a new random observation θ_i which is only dependent on θ_{i-1}. Thus it is in fact a Markov chain.

The transition kernel is given by

$$P(x, dy) = p(x, y)dy + r(x)\delta_x(dy) \tag{4.3}$$

as in equation (4.1) with $p(x, y) = q(y|x)\alpha(y, x)$.

We can show that this kernel fulfills the reversibility constraint.

$$\begin{aligned} \pi_t(x)p(x, y) &= \pi_t(x)q(y|x)\alpha(y, x) \\ &= \pi(x)q(y|x) \min\left(1, \frac{\pi_t(y)q(x|y)}{\pi_t(x)q(y|x)}\right) \\ &= \min\left(\pi_t(x)q(y|x), \pi_t(y)q(x|y)\right) \end{aligned}$$

which is symmetric and thus

$$\pi_t(x)p(x, y) = \pi_t(y)p(y, x)$$

Then from theorem 4.6 follows that θ is stationary with stationary distribution $\pi_t(\cdot)$. □

To show that the generated Markov chain indeed converges we want to use theorem 4.5. It states that irreducibility and aperiodicity of the Markov chain are sufficient to show its convergence. Irreducibility means that it is possible to move from x to dy in a finite number of steps with a positive probability for all states $x, y \in E$. Aperiodicity means that this number of steps is not forced to be a multiple of a fixed integer. Whether these conditions are fulfilled depends on the proposal density $q(\cdot|\cdot)$ (see equation (4.3) of the generated transition kernel). [11] states that the conditions are fulfilled if $q(\cdot|\cdot)$ has a positive density on the support of $\pi_t(\cdot)$. A random walk (as we will use later) fulfills the claimed conditions.

4.4 The Multilevel Delayed Acceptance algorithm

The multilevel delayed acceptance MCMC (MLDA) algorithm was introduced in [37] as a combination of Multilevel MCMC (MLMCMC) (see [15]) and Delayed Acceptance MCMC (DA) (see [12]). The goal is to decrease the computational costs and to overcome problems occurring in the MCMC algorithm, for example a high autocorrelation of the samples or a low stability of the algorithm.

MLMCMC is a sampling method using a hierarchy of levels - where we assume the lowest costs as well as the lowest accuracy on the coarsest level and accordingly the highest costs and highest accuracy on the finest level. This is an assumption which fits most models, because accuracy versus computational effort is a common trade-off. The main idea of MLMCMC is to make extensive computations on the coarse levels to have low computational costs and only few computations on the finest level to improve the accuracy of the result.

4.4.1 The Delayed Acceptance algorithm

The basic idea of DA is to use two different densities: the coarse density π_C, which is an approximation of the target density but needs less computational effort and the fine density π_F, which is the target density. The algorithm is described in algorithm 4.2. It consists of two steps. First, the Metropolis Hastings algorithm (see Algorithm 4.1) is executed using the coarse density to generate a proposal ψ. If it has been rejected in the MH algorithm and thus $\psi = \theta^{i-1}$ then we do not go to the second step but break the algorithm and set $\theta_i = \theta_{i-1}$. If ψ was accepted we compute again an acceptance probability, this time using the fine density π_F by which we decide whether we accept or reject the candidate ψ.

Algorithm 4.2 Delayed acceptance

```
function
$(\theta_1, ..., \theta_N) = \mathrm{DA}(\pi_F, \pi_C, q, \theta_0, N)$

Input
    target (fine) density $\pi_F(\cdot)$
    surrogate (coarse) density $\pi_C(\cdot)$
    proposal density $q(\cdot|\cdot)$
    starting point $\theta_0$
    number of steps $N$

Output
list of samples $(\theta_1, ..., \theta_N)$

Algorithm
  for i = 1,...,N do
    Generate a new proposal using the standard Metropolis-Hastings algorithm (algorithm 4.1)
        $\psi = MH(\pi_C, q(\cdot|\cdot), \theta_{i-1}, 1).$

    if $\theta_{i-1} = \psi$ then set $\theta_i = \theta_{i-1}$
    else
      Compute acceptance probability
        $\alpha(\psi, \theta_{i-1}) = \min\left(1, \frac{\pi_F(\psi) q_C(\theta_{i-1}|\psi)}{\pi_F(\theta_{i-1}) q_C(\psi|\theta_{i-1})}\right)$
      with $q_C(y|x) = \alpha_{MH}(x, y) q(y|x)$ where $\alpha_{MH}$ denotes the acceptance probability from the
      MH algorithm.

      Generate an equal distributed random number $u \in [0, 1]$.

      if $u \leq \alpha$ then accept candidate $\psi$ and set $\theta_i = \psi$.
      else reject candidate $\psi$ and set $\theta_i = \theta_{i-1}$
```

Remark

(i) Note that actually $q_C(y|x)$ is chosen as the proposal distribution generated in the MH algorithm in the first step

$$q_C(y|x) = \alpha_{MH}(x, y) q(y|x) + (1 - r(x)) \delta_x(y)$$

with $r(x) = 1 - \int_E p(x, y) dy$, but since the acceptance probability is only computed for $\theta_{i-1} \neq \psi$ the second term is always 0 [12].

(ii) In the original algorithm the coarse posterior distribution π_C is allowed to depend on the current state of the chain, but we assume it to be state-independent, because there is no proof of the stationarity of the following algorithms that are extensions of the DA algorithm for the state-dependent case yet. The DA algorithm coincides then with the so called surrogate transition method introduced in [35].

Theorem 4.8. *The DA algorithm described in algorithm 4.2 generates a Markov chain that is in detailed balance with the target density π_F.*

Proof
First note that the DA algorithm generates in each iteration a new random observation θ_i which is only dependent on θ_{i-1}. Thus it is in fact a Markov chain.

The transition kernel is given by

$$P(x, dy) = p(x, y)dy + r(x)\delta_x(dy)$$

as in equation (4.1) with $p(x, y) = \alpha(x, y)q_C(y|x)$.

Analogous to the proof of theorem 4.7 follows that

$$\pi_F(x)p(x, y) = \pi_F(y)p(y, x).$$

Thus the transition kernel $P(x, dx)$ fulfills the reversibility constraint and again from theorem 4.2 follows that θ is stationary with stationary distribution $\pi_F(\cdot)$. □

Remark
From the fact that q_C is in detailed balance with π_C follows that

$$q_C(x|y) = \frac{q_C(y|x)\pi_C(x)}{\pi_C(y)}$$

almost everywhere and thus the acceptance probability can be written as

$$\alpha(y, x) = \min\left(1, \frac{\pi_F(y)q_C(x|y)}{\pi_F(x)q_C(y|x)}\right) = \min\left(1, \frac{\pi_F(y)\pi_C(x)}{\pi_F(x)\pi_C(y)}\right). \tag{4.4}$$

For a discussion of the convergence we refer to [12]. Basically the assertion is, that if the MH algorithm used in the first step generates an irreducible and aperiodic chain these properties can be extended to the chain generated by the DA algorithm and then the convergence follows.

The DA algorithm can lead to a reduction of costs compared to the standard MH algorithm, because samples rejected at the coarse (cheap) level do not have to be evaluated at the fine (expensive) level. Furthermore it can be seen as an improvement of the proposal since the DA algorithm modifies the proposal kernel by executing the additional accept/reject-step to give effective proposals.

MLDA extends the DA algorithm in two ways: horizontally by replacing the single sample drawn at the coarse level by a complete sample chain and vertically by applying DA recursively on more than two levels. In the following we will expand the DA algorithm step by step.

4.4.2 The Randomized-Length-Subchain Surrogate Transition algorithm

The horizontally extended DA algorithm is called Randomized-Length-Subchain Surrogate Transition (RST) and can be found in alogrithm 4.3. The main idea is to not only execute one accept/reject-step but to run a short subchain consisting of multiple steps and use the last generated state as proposal for the fine level. The length of these subchains is also modeled as a random variable. Since it is an integer it is defined by a probability mass function which is given as an input argument into the RST algorithm. For example a minimum and a maximum subchain length $J_{min} \in \mathbb{N}$ and $J_{max} \in \mathbb{N}$ can be set and then a discrete uniform distribution over the values $\{J_{min}, ..., J_{max}\}$ is used to draw the subchain length. Note that another special case included in the algorithm is a deterministic choice of the subchain length.

Algorithm 4.3 Randomized-Length-Subchain Surrogate Transition

```
function
$(\theta_1, ..., \theta_N) = \mathrm{RST}(\pi_F, \pi_C, q, p, \theta_0, N)$

Input
    target (fine) density $\pi_F(\cdot)$
    surrogate (coarse) density $\pi_C(\cdot)$
    proposal density $q(\cdot|\cdot)$
    probability mass function $p(\cdot)$ for subchain lengths
    starting point $\theta_0$
    number of steps $N$

Output
list of samples $(\theta_1, ..., \theta_N)$

Algorithm
    for i = 1,...,N do
        Draw the subchain length $n \sim p(\cdot)$.
        Generate a subchain of length $n$ using the standard Metropolis-Hastings algorithm (algorithm 4.1)
            $(\psi_1, ..., \psi_n) = MH(\pi_C, q(\cdot|\cdot), \theta_{i-1}, n)$        (4.5)
        and set $\psi = \psi_n$ as proposal.

        Compute acceptance probability
            $\alpha(\psi|\theta_{i-1}) = \min\left(1, \frac{\pi_F(\psi)\pi_C(\theta_{i-1})}{\pi_F(\theta_{i-1})\pi_C(\psi)}\right).$        (4.6)

        Generate an equal distributed random number $u \in [0, 1]$.

        if $u \leq \alpha$ then accept candidate $\psi$ and set $\theta_i = \psi$.
        else reject candidate $\psi$ and set $\theta_i = \theta_{i-1}$
```

Theorem 4.9. *The RST algorithm in algorithm 4.3 generates a Markov chain that is in detailed balance with the target density π_F.*

Proof
First note that the RST algorithm generates in each iteration a new random observation θ_i which is only dependent on θ_{i-1}. Thus it is in fact a Markov chain.

The transition kernel of the subchain generated in equation (4.5) is given by

$$P_{sub}(x, dy) = \sum_{n\in\mathbb{Z}} p(n)P^n_{MH}(x, dy)$$

where $P^n_{MH}(x, dy)$ is the transition kernel of the Markov chain $P_{MH}(x, dy)$ generated by the standard Metropolis-Hastings algorithm (algorithm 4.1) n-times composed with itself.

Since P_{MH} obviously commutes with itself and we have shown in the proof of theorem 4.7 that it is reversible with respect to π_C follows by induction using theorem 4.3 that P_{sub} is also reversible with respect to π_C.

We can use equation (4.4) to show that the acceptance probability in equation (4.6) is similar to that in the DA algorithm (algorithm 4.2). Thus analogous to the proof of theorem 4.8 follows that it is in detailed balance with π_F.

$\square$

4.4.3 The Multilevel Delayed Acceptance algorithm

If we now extend the RST algorithm by introducing more levels we get the MLDA algorithm, it is denoted in Algorithm 4.4.

Algorithm 4.4 Multilevel delayed acceptance MCMC

```
function
$(\theta_1^l, ..., \theta_N^l) = \mathrm{MLDA}(l, \{\pi_i\}_{i=0}^l, q_0, \{p_i\}_{i=0}^l, \theta_0^l, N)$

Input
    number of levels $l$
    target densities $\pi_0(\cdot), ..., \pi_l(\cdot)$
    proposal density $q_0(\cdot|\cdot)$
    probability mass function $p_1(\cdot), ..., p_l(\cdot)$ for subchain lengths
    starting point $\theta_0^l$
    number of samples $N$

Output
list of samples $(\theta_1^l, ..., \theta_N^l)$

Algorithm
  for j = 1,...,N do
    Draw the subchain length $n \sim p_l(\cdot)$.

    Generate a subchain of length $n$ by executing the following step:
      if $l = 1$ then use the standard Metropolis-Hastings algorithm (algorithm 4.1)
          $(\psi_1, ..., \psi_n) = MH(\pi_0, q_0(\cdot|\cdot), \theta_{j-1}^l, n)$
      else call the MLDA algorithm recursively for $l - 1$
          $(\psi_1, ..., \psi_n) = MLDA(l-1, \{\pi_i\}_{i=0}^{l-1}, q_0, \{p_i\}_{i=0}^{l-1}, \theta_{j-1}^{l-1}, n).$   (4.7)

    Set the new coarse proposal $\psi = \psi_n$.

    Compute the acceptance probability
          $\alpha(\psi, \theta_{j-1}^l) = \min\left(1, \frac{\pi_l(\psi)\pi_{l-1}(\theta_{j-1}^l)}{\pi_l(\theta_{j-1}^l)\pi_{l-1}(\psi)}\right).$   (4.8)

    Generate an equal distributed random number $u \in [0, 1]$.

    if $u \leq \alpha$ then accept candidate $\psi$ and set $\theta_j^l = \psi$.
    else reject candidate $\psi$ and set $\theta_j^l = \theta_{j-1}^l$
```

Remark
We simplified the original algorithm introduced in [37] in the following point that is not relevant in our application.

While in algorithm 4.4 the states in the fine and coarse density are the same, the original algorithm allows them to differ. In this case it is necessary to add the proposal densities $q_{1,F}(\cdot|\cdot), ..., q_{l,F}(\cdot|\cdot)$ as input and to draw the fine-mode proposal $\psi_F \sim q_{l,F}(\cdot|\theta^j_{l,F})$.

Theorem 4.10. *The MLDA algorithm in 4.4 generates a Markov chain that is in detailed balance with the target density π_l.*

Proof
We proof this theorem by induction over the number of levels. The base case is the DA algorithm (algorithm 4.2), because for $l = 1$ the MLDA algorithm is similar to the DA algorithm for which we have already proven that the generated chain is in detailed balance with the target density.

Let us now assume that the MLDA algorithm for a fixed number of levels l generates a Markov chain that is in detailed balance with π_l. If we then run the algorithm for $l + 1$ levels we know that the chain generated in the recursive call of the MLDA algorithm with l levels (equation (4.7)) is in detailed balance with the target density π_l.

Again we can use equation (4.4) to show that the acceptance probability $\alpha(\psi, \theta^{l+1}_{j-1})$ defined in equation (4.8) is similar to that in the DA algorithm (algorithm 4.2). Thus analogous to the proof of theorem 4.8 follows that the generated Markov chain is in detailed balance with π_L.

□

4.5 Evaluation methods

MCMC algorithms are used to simulate a target distribution by generating a set of samples and using the empirical distribution of them as an approximation. While we have shown that the Markov chains generated by the introduced MCMC algorithms are converging to the target distribution, we have no statement concerning the convergence rate. Though, if we want to apply these algorithms the performance and the costs get relevant. On the one hand we want to run the algorithm for a number of steps large enough to generate a chain that is a good approximation of the target distribution, but on the other hand the number of steps should be as low as possible to reduce the computational costs. For that reason it is important to have diagnostic methods to decide whether a sample fulfills these requirements and to spot and specify occurring problems.

To find good diagnostic methods it is useful to first have a look at the problems that might occur. According to [53] there are two main problems:

1. The rate of samples from distributions that are different from the target distribution is high.
2. The samples show a high serial correlation, which is the case either if we accept only few candidates and thus have many equal samples or if the proposal distribution does not walk consistent through the full domain of the target distribution but remains long in a small region of it.

If we spot one of these problems we have different possible strategies to solve them. The first problem we can try to fix by increasing the burn-in size. An idea to decrease the serial correlation is to adjust the parameters of the MCMC algorithm. Both problems can usually be handled as well by increasing the sample size, but this will increase the runtime of the algorithm as well.

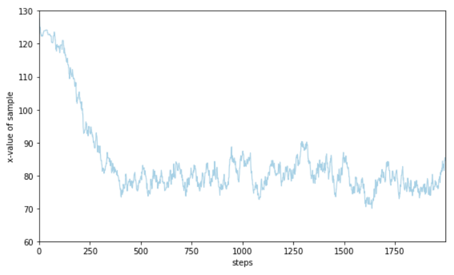
In the following we want to give some concrete evaluation methods. First we explain possibilities of a graphical evaluation. Usually this shows issues in the generated sample and indicates possible reasons for a bad performance. Afterwards we focus on the computation performance indicators that evaluate the quality of the whole sample in one concrete value. This can be used to find problematic samples in a large number of generated chains or to compare different algorithms.

4.5.1 Graphical evaluation

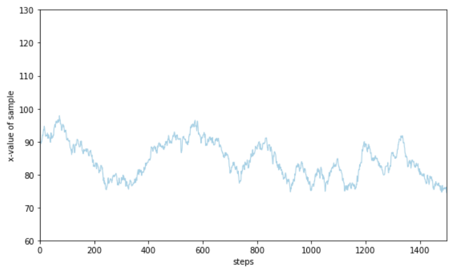
First it is important to mention, that the following methods analyze the generated chains in each dimension seperately. Though they can be applied for each dimension, it can be sufficient to analyze only a selection of the dimensions - especially in high-dimensional applications, because this is often enough to spot occurring problems.

The most basic evaluation method is the trace plot which is a line chart that plots the steps at the x-axis and the value of the according sample (in one dimension) at the y-axis. Trace plots give a good overview about the generated sample and help to find problems like a too small burn-in or a high serial correlation of successive samples.

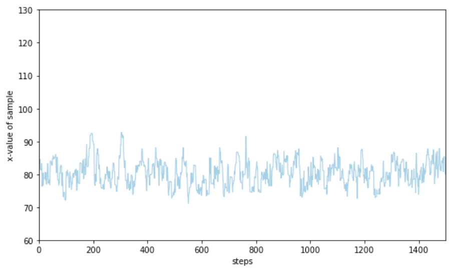
In figure 4.1 there are three example trace plots to illustrate how those plots can be used for diagnostics. In figure 4.1a one can clearly see that the algorithm needs a number of steps to move from the starting point to the real distribution. We could fix this problem easily by increasing the burn-in. Another problem can be spotted in figure 4.1b. In this example we observe a high correlation between successive draws. We could try to fix this by changing parameters of the algorithm. In figure 4.1c we see a trace plot of a sample where we can't spot any problem.



(a) Trace plot of a sample with a too small burn-in size.



(b) Trace plot of a sample with a high serial correlation.



(c) Trace plot not showing any problems.

Figure 4.1: Different trace plots visualizing the insights offered by the trace plot.

While the trace plot gives us a first sense for the correlation of samples we can analyze it more precisely computing the autocorrelation. The mathematical definition of the autocorrelation that we use can be found in [52]. Given a sequence of random variables $X_1, X_2, ..., X_N$ the autocorrelation coefficient can be defined as

$$\rho(n,k) = \frac{Cov(X_n, X_{n+k})}{\sqrt{Var(X_n)Var(X_{n+k})}}$$

for all $n = 1, 2, ..., N$ and $k = 1, 2, ..., N - k$. It is a measure of the serial correlation of variable n and the following k variables. If the random variables $X_1, X_2, ...$ are drawn from the same distribution, the autocorrelation coefficient is independent of n and we call

$$\rho_k = \frac{Cov(X_1, X_{k+1})}{\sqrt{Var(X_1)Var(X_{k+1})}}$$

the autocorrelation at lag k.

However, we do not have random variables given but a set of samples. Assume we have N samples $\theta_1, ..., \theta_N$. Following [52] we can now use these samples to compute an estimator of ρ_k by

$$\hat{\rho_k} = \frac{\frac{1}{N-k}\sum_{n=1}^{N-k}(\theta_n - \hat{\mu})(\theta_{n+k} - \hat{\mu})}{\frac{1}{N}\sum_{n=1}^{N}(\theta_n - \hat{\mu})^2}$$

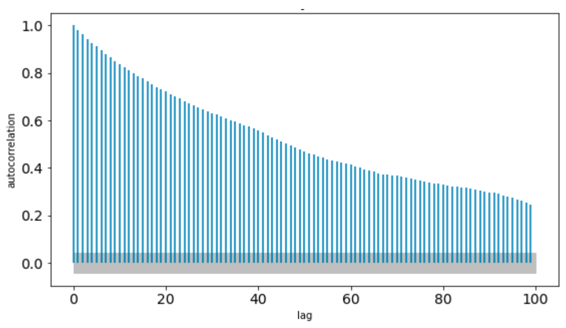
with sample mean

$$\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N}\theta_n. \tag{4.9}$$

We can plot the autocorrelation as a bar chart having the lag at the x-axis and the corresponding autocorrelation estimator at the y-axis. Two example plots can be found in figure 4.2.

4.5.2 Performance indicators

As the graphical evaluation methods introduced before, the here presented performance indicators are applied for each dimension apart. To get one value for all dimensions together depending on the indicator the minimum,



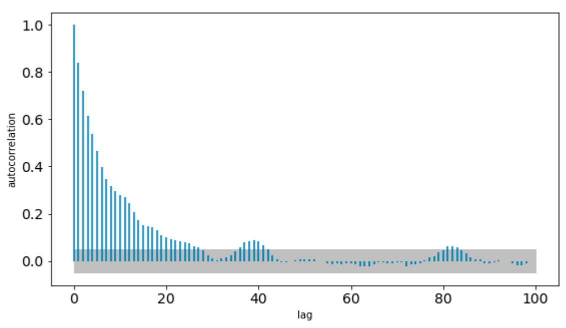


Figure 4.2: Autocorrelation plots of two different samples; the left sample has a slowly decreasing autocorrelation. A sample is still markable correlated with the sample generated after 100 steps of the algorithm. The right plot shows a case where the autocorrelation is decreasing fast. After a few steps of the algorithm we generate a sample that is completely uncorrelated with the current one. This indicates that the algorithm that has generated the right sample will be more efficient and will show a higher stability.

maximum or mean over all dimensions can be used.

A first idea is to put as much information as possible from the autocorrelation plot into one performance indicator by summing up the autocorrelations for lag $k = 1, 2, \ldots$. Though, with increasing k the noise in the autocorrelation $\hat{\rho}_k$ increases. Thus we truncate the sum at $K = \max(k \in \mathbb{N} \mid \hat{P}_t > 0 \; \forall t = 1, \ldots, k)$ and define the integrated autocorrelation time

$$\hat{\tau} = -1 + 2 \cdot \sum_{t=0}^{K} \hat{P}_t \tag{4.10}$$

for $\hat{P}_t = \hat{\rho}_{2t} + \hat{\rho}_{2t+1}$.

Another diagnostic measure is the effective sample size, which describes the number of independent observations forming an equivalent sample. That means if we have a large sample size N but the decay of the correlation between the terms of the Markov chain is slow, we still have a small effective sample size. The effective sample size is given by

$$N_{eff} = \frac{N}{1 + 2 \cdot \sum_{t=1}^{\infty} \rho_t}$$

(see [56]). We compute an estimator of the effective sample size by using the estimator of the autocorrelation $\hat{\rho}_k$ and truncating the sum again. The resulting estimator is given by

$$\hat{N}_{eff} = \frac{N}{\hat{\tau}}$$

with integrated autocorrelation time $\hat{\tau}$ defined as in equation (4.10).

Besides the introduced measures and plots there exists a general strategy to improve the diagnostic results, that is taking into account multiple sample sets and not only one. This can be done in two ways:

1. **Split sample set:** A method to get a general idea of the quality of a set of samples is splitting this set into subsets, called chunks, and analyze each of them. If we get significantly different results, provided the chunks have a sufficient size, it indicates that there is a problem. On the one hand we can compute statistics as the mean, the variance or higher moments. On the other hand we can compare trace and autocorrelation plots.

2. **Multiple chains:** A similar idea as splitting the sample set is to run the chain multiple times and comparing the results. It is usually a good idea to start the chains from different points, too (see [56]).

We assume that we have generated M chains (or chunks) each of length N. For $n = 1, \ldots, N$ and $m = 1, \ldots, M$ let θ_n^m denote sample n of chain (chunk) m. Now we can adapt the definitions of autocorrelation and effective samples size we introduced before.

We define the within-chain variance estimator

$$W = \frac{1}{M} \sum_{m=1}^{M} s_m^2$$

with $s_m^2 = \frac{1}{N-1} \sum_{n=0}^{N} (\theta_n^m - \hat{\mu}_m)^2$ where $\hat{\mu}_m = \frac{1}{N} \sum_{n=1}^{N} \theta_n^m$ is the chain mean.

Furthermore we define the between-chain variance estimator

$$B = \frac{N}{M-1} \sum_{m=1}^{M} (\hat{\mu}_m - \hat{\mu})^2$$

with sample mean $\hat{\mu}$ defined as in equation (4.9).

The variance estimator is then defined as a combination of the within-chain and the between-chain variance:

$$v\hat{a}r^{+} = \frac{N-1}{N} W + \frac{1}{N} B.$$

Now we can compute the autocorrelation as

$$\hat{\rho}_k = 1 - \frac{W - \sum_{m=1}^{M} \hat{\rho}_{m,k}}{v\hat{a}r^{+}} \tag{4.11}$$

where $\hat{\rho}_{m,k}$ denotes the autocorrelation of a chain (chunk) given by

$$\hat{\rho_{m,k}} = \frac{\frac{1}{N-k} \sum_{n=1}^{N-k} (\theta_n^m - \hat{\mu})(\theta_{n+k}^m - \hat{\mu}_m)}{\frac{1}{N} \sum_{n=1}^{N} (\theta_n^m - \hat{\mu}_m)^2}.$$

Putting all this together results in two measures we will use to evaluate and compare different algorithms. Their definitions are given in the following.

Definition 4.10 (Integrated autocorrelation time)**.** *The integrated autocorrelation time (IAT) is given by*

$$\hat{\tau} = -1 + 2 \cdot \sum_{t=0}^{K} \hat{P}_t$$

for $\hat{P}_t = \hat{\rho}_{2t} + \hat{\rho}_{2t+1}$ *and* $\hat{\rho}_t$ *as in equation* (4.11).

Definition 4.11 (Effective sample size)**.** *The effective sample size (ESS) is given by*

$$\hat{N}_{eff} = \frac{NM}{\hat{\tau}}.$$

5 Application of MCMC algorithms to reconstruct a source distribution

In the previous chapters we explained the problem of EEG source reconstruction on the one hand and introduced the mathematical theory of MCMC algorithms on the other hand. At this point we are ready to put everything together and apply the introduced algorithms to the EEG inverse problem. We pursue this task in a practical way by implementing a program applying the MH and the MLDA algorithm to reconstruct a source distribution of EEG measurement values.

The aim of this chapter is to explain the experimental setup, including the construction of the head model, the standardization of the sensor values and a discussion of the restrictions we make. Furthermore we expound the concrete design of the MH and the MLDA algorithm as we use them in our implementation. In particular, we will discuss the construction of different hierarchies for the MLDA algorithm. Finally we will give a brief introduction in the architecture of the implementation and the used software.

5.1 Experimental setup

First of all, it should be stated that in the major part of our numerical experiments we will use a two-dimensional head model instead of a three-dimensional model for the following two reasons.

1. We have a lower computation time which allows to run a number of different experiments with a high number of samples.
2. The visualization of two-dimensional results is straightforward and thus insights in the behavior of the algorithms can easily be gained.

Hence, we explain the settings for the two- and the three-dimensional case in the following.

Before diving into the details of the experimental settings, let us clarify the goal of the reconstruction of the source distribution. Given measurement values, that we assume to be with noise of a certain distribution, we try to reconstruct the distribution of the dipole causing the measured potentials. This dipole is defined by its position and its orientation and is denoted by the random variable

$$S = (X, Y, Z, R, \Phi) \in [0, 256] \times [0, 256] \times [0, 256] \times [0, 2\pi) \times [0, 2\pi)$$

in the three-dimensional case and

$$S = (X, Y, R) \in [0, 256] \times [0, 256] \times [0, 2\pi)$$

in the two-dimensional case.

We make a number of assumptions and restrictions regarding the dipole. First of all, we do not reconstruct the strength of the dipole, but assume that it has unit-strength. However, an adaption of the model including the strength is possible and could be discussed in further studies. Furthermore we assume that the source of the measured electric potential is a single dipole. This restriction is necessary to simplify the problem. Finally, we assume the dipole to be located in the gray matter. This assumption is physiologically plausible (see [57]).

Basis of the source reconstruction is a head model that allocates the different compartments. We use a 5-compartment model consisting of white matter, gray matter, CSF, skull and scalp. We assume constant conductivity for each head compartment and use standard values from literature listed in table 5.1. Our head model is created by using a tissue probability map to construct a hexahedral mesh. The tissue probability map is generated by MRI and adjusted by multiple steps. It splits the domain in $256 \times 256 \times 256$ cells of equal size and assigns to each of these voxels a probability for each tissue $P_{i,j}(\text{tissue})$. Using these tissue probabilities a $256 \times 256 \times 256$ mesh can

Compartement	Conductivity (S/m)
White matter	0.14
Gray matter	0.33
CSF	1.79
Skull	0.01
Skin	0.33

Table 5.1: Standard tissue conductivities (see [59])

be created by assigning a tissue to each cell using the following logic:

$$\text{tissue(i,j,k)} = \begin{cases} \text{white matter} & \text{if } P_{i,j,k}(\text{white matter}) > 0.5 \\ \text{gray matter} & \text{if } P_{i,j,k}(\text{white matter}) + P_{i,j,k}(\text{gray matter}) > 0.5 \\ \text{CSF} & \text{if } P_{i,j,k}(\text{white matter}) + P_{i,j,k}(\text{gray matter}) + P_{i,j,k}(\text{CSF}) > 0.5 \\ \text{skull} & \text{if } P_{i,j,k}(\text{white matter}) + P_{i,j,k}(\text{gray matter}) + P_{i,j,k}(\text{CSF}) \\ & \quad +P_{i,j,k}(\text{skull}) > 0.5 \\ \text{scalp} & \text{if } P_{i,j,k}(\text{white matter}) + P_{i,j,k}(\text{gray matter}) + P_{i,j,k}(\text{CSF}) \\ & \quad +P_{i,j,k}(\text{skull}) + P_{i,j,k}(\text{scalp}) > 0.5 \\ \text{none} & \text{else.} \end{cases} \tag{5.1}$$

This mesh contains all cells, even the ones lying outside the head domain. We call it the full mesh and keep the cells ordered. For a given point $p \in [0, 256]^3$ we can easily determine the index of the cell, the point lies into.

To solve the forward problem we use reduced meshes, where all cells lying outside the head - that means no tissue is assigned - are removed since they are not part of our domain. We have a reduction vector v of length $(256 + 1)^3$ that contains a 1 at position i if node i of the full mesh is contained in the reduced mesh as well and a 0 otherwise. The five compartments of the three-dimensional head model are displayed in figure 5.1. To measure the potential differences there are placed 69 electrodes on the scalp.

As a two-dimensional head model we use a sagittal slice of the three-dimensional head model (see figure 5.2) with the same constant conductivity values as in the three-dimensional model. We set 36 electrodes on the scalp positioned as shown in figure 5.2.

Having the reduced mesh and the electrode positions we can compute the transfer matrix as explained in chapter 2. It can be used each time we solve the forward problem for this setting.

An important point to consider is the standardization of the sensor values $b \in \mathbb{R}^M$, concerning the measured as well as the simulated values. EEG measures only potential differences, but the absolute values can be shifted arbitrarily. There exist different options of a standardization. The first option is to choose a reference electrode and shift the measured potential differences by the value measured at the reference electrode. That means each value corresponds to the difference of the potential measured at this electrode and the potential measured at the reference electrode. The value at the $i-$th sensor is then given by

$$\tilde{b}_i = b_i - b_0,$$

assuming sensor 0 to be the reference. Another option is to set the mean over all potential differences to zero. Thus we subtract the mean of all measured values from each potential difference and get a centered vector. The value at the $i-$th sensor is then given by

$$\tilde{b}_i = b_i - \frac{1}{M}\sum_{j=0}^{M} b_j.$$

The latter option is the one we use through the following numerical experiments. Furthermore we have to normalize the potential differences since we do not consider the strength of the source dipole but only its position and orientation. For this reason we divide each potential by the L2-norm of the whole potential vector and get the

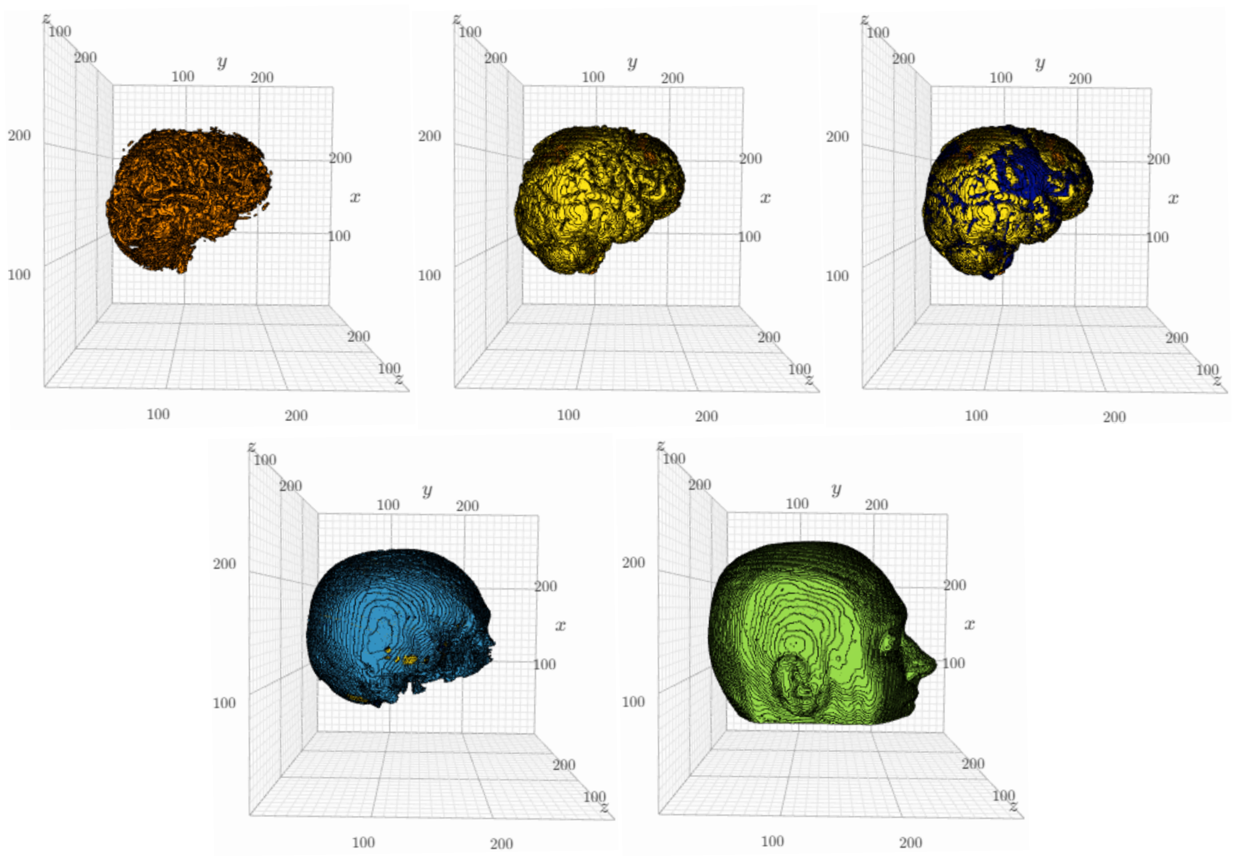


Figure 5.1: The five compartments of the three-dimensional head model that is used in the experiments. From the upper left to the lower right image in each picture one more compartment is displayed, starting from the inner of the brain with the white matter (orange) followed by gray matter (yellow), CSF (dark blue), skull (light blue) and scalp (green).

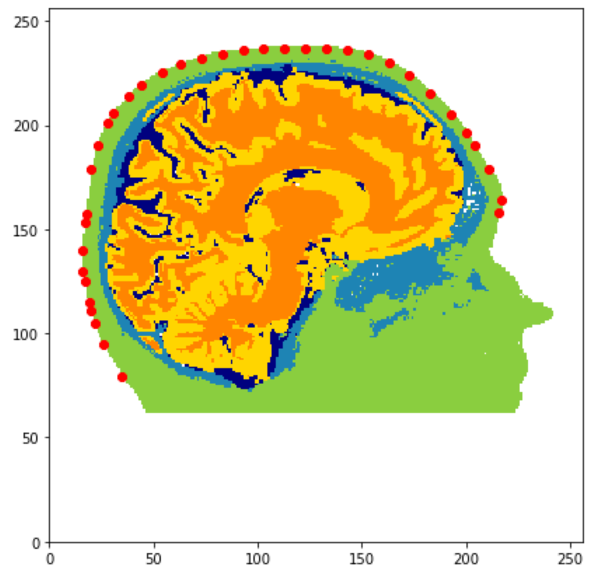


Figure 5.2: two-dimensional head model that is used in the experiments consisting of the five compartments white matter (orange), gray matter (yellow), CSF (dark blue), skull (light blue) and scalp (green) with 36 electrodes positioned on the scalp (red dots).

normalized, standardized value

$$\hat{b}_i = \frac{\tilde{b}_i}{||\tilde{b}||_{L2}}$$

for the $i-$th sensor value, which we can use for comparisons.

5.2 Metropolis Hastings algorithm

To use the MH algorithm described in algorithm 4.1, we have to set all required arguments first. This involves the proposal density $q(\cdot|\cdot)$, the target density π_t and a starting point θ_0.

As proposal density, which is the density used to generate the candidates, we use the density function of a normal distribution

$$q(x|y) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(x-y)^T \Sigma^{-1}(x-y)\right) \tag{5.2}$$

with $d = 3$ and diagonal covariance matrix $\Sigma = diag(\sigma_x, \sigma_y, \sigma_\rho)$, in the two-dimensional setting, and $d = 5$ and $\Sigma = diag(\sigma_x, \sigma_y, \sigma_z, \sigma_\rho, \sigma_\varphi)$ in the three-dimensional case, where σ_x and σ_y (and σ_z) denote the variance in the position dimensions and σ_ρ (and σ_φ) describe(s) the variance in the orientation dimension(s). Concrete variance choices are discussed later when we present the results.

To compute the posterior distribution we want to use as much information about the dipole and the model as possible.

1. Most important we solve the forward problem for the candidate ψ using the transfer matrix, which means we compute the values b we expect to measure at the electrodes if ψ was our dipole. This solution can be compared with the actually measured values b_{ref}.
2. Since we assume the dipole position to be restricted on the gray matter, we use the gray matter tissue probability denoted by the function $\phi(\cdot)$ that is given by the tissue probability map of the MRI we have already used to construct the head model before. The confidence in this probability is reflected by the factor w, which we choose as $w = 10^{-3}$.

We define the target density as

$$\pi_t(b) = ((1-w)\phi(x) + w)\left(\frac{1}{\sqrt{(2\pi)^M |\Gamma|}} \exp\left(-\frac{1}{2}(b_{ref} - b)^T \Gamma^{-1}(b_{ref} - b)\right)\right) \tag{5.3}$$

with diagonal covariance matrix $\Gamma = diag(\gamma_1^2, ...\gamma_M^2)$.

To choose the start point different approaches are conceivable. A first option is the choice of a fixed start point, i.e. the center of the domain with orientation angles equal to zero. Furthermore a random dipole could be chosen to initialize the chain. Putting a bit more effort in the choice of the start point, the dipole showing the highest target density of a set of a fixed or randomly chosen dipoles could be chosen. This can decrease the convergence time of the algorithm.

5.3 MLDA algorithm

Before running the MLDA algorithm, the most important question is how we want to choose our levels and how many levels we want to use. Since the main idea is to have lower costs at the coarse level(s) we analyze the costs of the computation of the posterior density to get an idea of possible cost reductions.

Let us assume that we are in dimension D and have M electrodes and a mesh with N nodes. In each step of the MCMC algorithm, a candidate is generated and evaluated by its posterior. The major part of the costs arises from the computation of the posterior. This computation includes the following three steps: solving the forward problem, determining the tissue probability and computing π_t by its formula given in equation (5.3).

For the solution of the forward problem we first have to compute the right hand side (see chapter 2.4). Before the patch can be built we have to find the cell, which needs costs of size $\mathcal{O}(log(N))$. The further costs depend directly on the chosen source model. We use the Venant approach and Partial Integration. Regarding the costs, building the patch as well as assembling the vector have higher costs in the Venant approach by factor C^D where the constant C denotes the number of cells per dimension we include in the Venant-patch.

Furthermore we have to load the transfer matrix. The required loading time depends highly on the memory where the transfer matrix is stored. Currently the cache-RAM-ratio is approximately $1/10 - 1/100$ ([51]) and the cache size is $2 - 12MB$ ([47]). The transfer matrix has $M \cdot N$ entries and we need 8 bytes to store a double and 4 bytes for a float. Thus to store the full transfer matrix we need $8MN$ or $4MN$ bytes. Thus having $MN \leq 10^6$ would lead to an immense decrease of the loading time (by factor 10 to 100).

Finally we have to multiply the dense transfer matrix of size $M \times N$ with the sparse right hand side vector. The costs depend on the number of non-zeros NNZ contained in the right hand side vector. This is 2^D for Partial Integration and $(C+1)^D$ for the Venant approach. Thus we have costs of order $\mathcal{O}(\tau \cdot N^d \cdot M)$.

The costs of getting the tissue probability are given by the costs we have to find the required cell. These are in $\mathcal{O}(1)$ since we can use the full (structured) mesh here. To compute the posterior density π_t using the forward solution as in equation (5.3) we have costs of size $\mathcal{O}(M)$.

Step	**Costs**
Solving the forward problem	$\mathcal{O}(\log(N) + \tau \cdot NNZ \cdot M)$
Getting the tissue probability	$\mathcal{O}(1)$
Computing π_t	$\mathcal{O}(M)$

Table 5.2: Costs of the computation of the posterior density π_t for one sample.

A summary of the computational costs can be found in table 5.2. The resulting total costs are of order $\mathcal{O}(\log(N) + \tau \cdot NNZ \cdot M)$. These theoretical considerations show that the major part of the costs is caused by the matrix-vector multiplication in the step of solving the forward problem.

A quantification of the costs is difficult since it depends strongly on the implementation. Measuring the run time of the implemented computation of the posterior in the program code written for the numerical experiments in this thesis does not show any differences for varying the number of sensors, the mesh size or the source model. The step of solving the forward problem implemented in the used software DUNEuro also does not show any differences in the run time for varying numbers of sensors. A reason for this could be an inefficient implementation, resulting in a high overhead.

Nevertheless, we simulated in C++ the multiplication of a dense matrix with a sparse vector containing either 2^D entries to simulate the case of Partial integration or 4^D entries to simulate the choice of the Venant source model. The results that are relevant for our setup are shown in table 5.4. Table 5.3 gives an overview of the number of nodes in the different meshes as well as the size of the corresponding transfer matrices, which is the basis for the performance tests. We analyze the three-dimensional case here because it is the more realistic setting.

These performance tests show the following cost reductions.

1. A decrease of the number of non zeros in the sparse vector leads to a decrease of the computational costs by the same factor. Thus, using the Partial Integration instead of the Venant approach leads to a decrease of the costs of the matrix-vector product by factor 1/8.

2. A decrease of the number of sensors leads to a decrease of the computational costs by the same factor.

3. Regarding the effect of a decrease of the number of nodes N we observe a logarithmic decrease.

Size of full mesh	**Number of nodes in reduced mesh**	**Size of transfer matrix**		
		64 sensors	32 sensors	16 sensors
$256 \times 256 \times 256$	$4,053,267$	1413MB	706MB	353MB
$128 \times 128 \times 128$	$517,324$	177MB	88MB	44MB
$64 \times 64 \times 64$	$66,921$	22MB	11MB	5.5MB

Table 5.3: Number of nodes in the reduced meshes and corresponding transfer matrix sizes. The blue marked cells indicate a matrix that can be stored in the cache.

	Venant ($NNZ = 64$)			Partial Integration ($NNZ = 8$)		
	$N = 4 \cdot 10^6$	$N = 5 \cdot 10^5$	$N = 6.25 \cdot 10^4$	$N = 4 \cdot 10^6$	$N = 5 \cdot 10^5$	$N = 6.25 \cdot 10^4$
$M = 64$	$61\mu s$	$42\mu s$	$21\mu s$	$8\mu s$	$5\mu s$	$3\mu s$
$M = 32$	$30\mu s$	$19\mu s$	$7\mu s$	$4\mu s$	$2\mu s$	$1\mu s$
$M = 16$	$14\mu s$	$8\mu s$	$3\mu s$	$2\mu s$	$1\mu s$	$0.5\mu s$

Table 5.4: Run time of the matrix-vector multiplication of a dense matrix of size $M \times N$ and a sparse vector of size N with 64 respectively 8 non-zero entries. The blue marked cells indicate a matrix of size lower than 12MB.

4. We can see caching effects if the matrix size decreases 12MB. Thus if we compare the run time of a matrix with a size greater than 12MB and a matrix with a size smaller than 12MB the costs increase even further by a factor of approximately 1/2.

As a result of these observations we define different hierarchies above the following parameters, because they affect the computational costs:

- Number of electrodes
- Number of nodes contained in the mesh
- Source model

Hierarchy 1: Number of electrodes
Regarding the reduction of costs it seems promising to reduce the number of electrodes. There exist different possible strategies, how we can realize this reduction. Let us assume we have M electrodes and want to reduce the number to $K < M$.

A first idea is to choose K electrodes randomly. Since in realistic experimental setups there are usually only a few hundred electrodes, there is a high risk of selecting some electrodes that have a high correlation and loosing a high amount of relevant information. Another idea is to choose K electrodes by a geometrical clustering. In the two-dimensional case this could be a simple choice like selecting each K-th electrode as illustrated in figure 5.3. In the three-dimensional case this is a bit more complicated, but one could imagine a geometric partition of the head surface in K parts and the selection of one electrode per part.

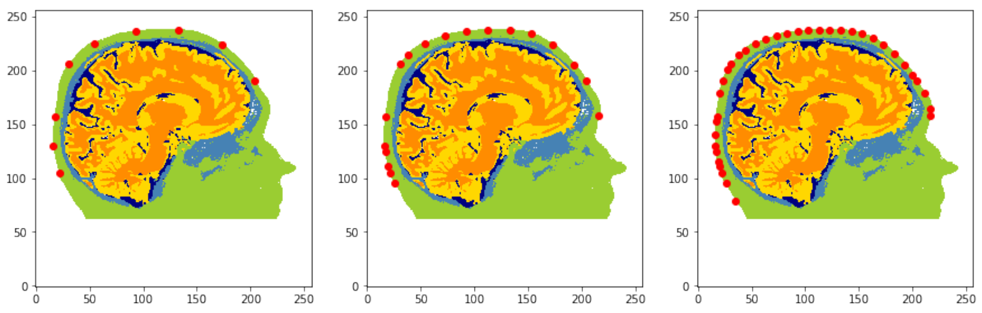


Figure 5.3: Possible reduction of the number of electrodes by selecting each K-th electrode.

Both ideas aim to choose K of the M existing electrodes, but this is not necessary. We can also choose M new positions. Adapting the idea of a geometrical clustering we could partition the head surface in K parts and choose the average position of all contained electrodes as the new electrode position.

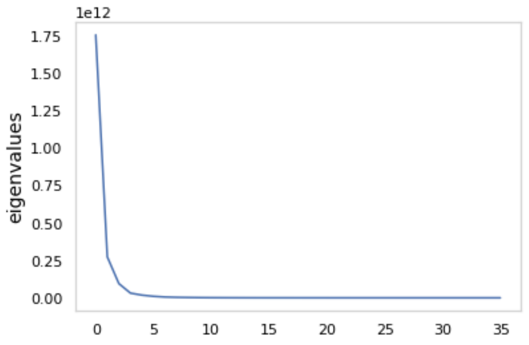
Going one step further, we do not even have to choose positions of electrodes, because for the solution of the forward problem using Partial Integration or the Venant approach, we only need the mesh, the transfer matrix and the measurement values. Thus, we can transform the transfer matrix as well as the measurement values with respect to the full number of electrodes by a linear combination onto a transfer matrix and corresponding measurement values in a lower dimension. To determine the linear coefficients in a way that keeps as much information as possible we apply a Principal Component Analysis (PCA). PCA is a widely used statistical method for dimension reduction. An introduction can be found for example in [33], mathematical details in [43]. The main idea is to transform the data by a linear orthogonal transformation onto a lower dimension by retaining as much information as possible. The new dimensions are called principal components. They can be constructed using a singular value decomposition of the covariance matrix. The first K eigenvectors are chosen as the principal components.

We start with the mean-centered transfer matrix $T \in \mathbb{R}^{M \times N}$, where M denotes the number of sensors and N the number of nodes. In the first step we compute the outer product of the transfer matrix

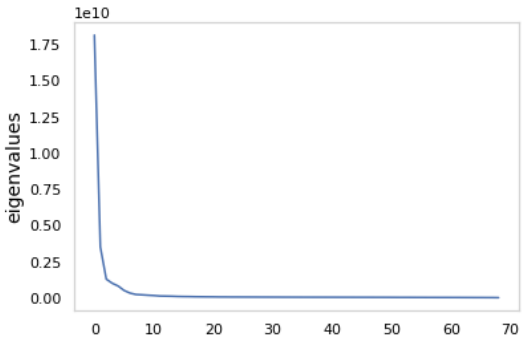
$$C = TT^T \in \mathbb{R}^{M \times M}$$

which indicates the covariance of the electrodes. The intuition behind this is that the matrix-vector product of the transfer matrix with a vector containing the source configuration gives us the values at the sensors. Hence, if we multiply the transfer matrix with its transposed matrix, we get a matrix containing entries $C_{i,j}$ indicating the potentials measured at sensor i when the dipole causing the potential is located at sensor j. We will justify this intuition later with real data (see chapter 7).

Then we compute the eigenvalues sorted in ascending order $\lambda_1, ...\lambda_M$ and the corresponding eigenvectors $v_1, ..., v_M$ of this matrix. Analyzing the eigenvalues we see a rapid decrease (Fig. 5.4). Computing the proportion of the sum of the first K electrodes at the sum of all eigenvalues shows, that in the two-dimensional case the first half of 18 eigenvalues covers 99,8% and the first quarter of 9 eigenvalues covers a portion of 99.1%. In the three-dimensional setting, we observe that the first half of 35 eigenvalues covers a portion of 97%, the first quarter of 17 eigenvalues still covers 95% of the total sum of eigenvalues. This indicates that we can reduce the number of sensors while keeping a high portion of information.



(a) Sorted eigenvalues in the two-dimensional case.



(b) Sorted eigenvalues in the three-dimensional case.

Figure 5.4: Decrease of the eigenvalues of the correlation matrix of the sensors, indicating that a reduction of sensors keeping a high portion of information is possible.

We set

$$V = [v_1, ..., v_K]^T \in \mathbb{R}^{K \times M}$$

as the matrix consisting of the first K eigenvectors and

$$\tilde{T} = VT \in \mathbb{R}^{K \times N}$$

as the new transfer matrix.

We stated before, that the outer product of the transfer matrix indicates the covariance of the sensors. Therefore, we get the correlation between the sensors if we normalize it row wise with its diagonal entries. The correlation of the new sensors (Fig. 5.5) shows, that the new sensors are uncorrelated and thus keep as much information as possible.

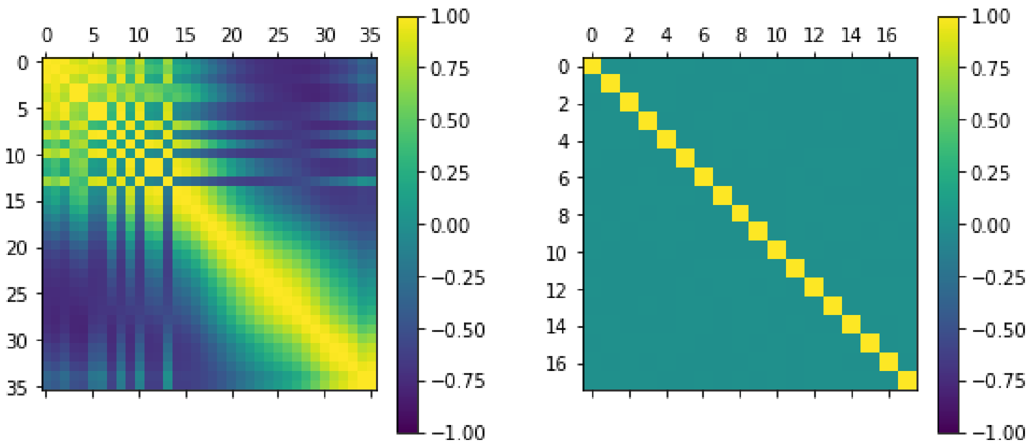


Figure 5.5: Correlation matrices of the 36 physical sensors (left) and of 18 virtual sensors constructed by a PCA as described in the text (right), showing that there is a high correlation of the measured potentials at the physical sensors, especially between neighboring sensors. Though, the transformation by PCA has constructed fully uncorrelated sensors, which is important for keeping as much information as possible.

For a right hand side l we can compute the values b at the M electrodes by $b = Tl \in \mathbb{R}^M$ and the values at the K virtual electrodes by

$$\tilde{b} = \tilde{T}l = VTl = Vb \in \mathbb{R}^K.$$

Thus, instead of a transfer matrix of size $M \times N$ we now have a transfer matrix of size $K \times N$.

Since we use the variances of the sensors in the posterior we have to transform them as well to get the variances of the new virtual electrodes. Note, that we assume the variances of the measured values at the sensors to be given. Computing the correlation of the sensors using the transfer matrix as stated before, we can multiply this correlation matrix row wise by the given variances to get a covariance matrix containing the given variances in its diagonal. This covariance matrix, let us denote it by D can be used to compute the covariances of the constructed virtual sensors by computing

$$\tilde{D} = VDV^T$$

containing the variances of the new sensors in its diagonal.

Hierarchy 2: Mesh
If we want to reduce the number of nodes contained in our mesh, we have to increase the cell width. We create two new coarse meshes, a full and a reduced one. To create the full coarse mesh we combine cells and compute the tissue probabilities of the new cell as the average of the tissue probabilities of the combined cells. For example the mesh of size 128×128 is created by computing tissue probabilities for the new cells by

$$P^{128}_{i,j}(\text{tissue}) = (P_{2i,2j}(\text{tissue}) + P_{2i+1,2j}(\text{tissue}) + P_{2i,2j+1}(\text{tissue}) + P_{2i+1,2j+1}(\text{tissue}))/4$$

for $i, j = 0, ..., 127$ and following equation ((5.1)). The reduced coarse mesh is again generated by removing all cells with no assigned tissue, thus we get again a reduction vector v_c. Analogously coarser meshes can be constructed. Fig. 5.6 shows the different meshes for the two-dimensional setting.

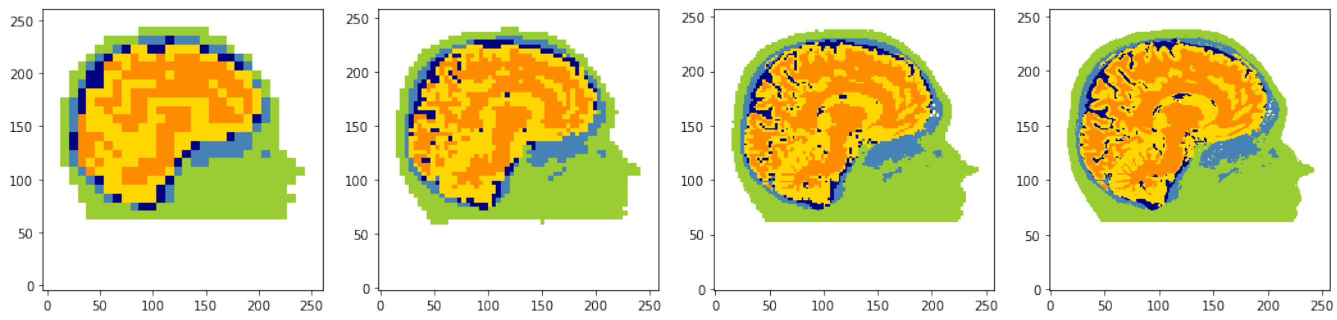


Figure 5.6: Different coarse two-dimensional meshes: 32×32, 64×64, 128×128 and 256×256 (from left to right).

We have to determine new transfer matrices for the new meshes. The key idea is to not compute them on the coarse mesh, but to transform the transfer matrix from the fine mesh similar as we did before when reducing the number of electrodes. The idea is to represent each node of the coarse mesh as a linear combination of the values at the neighbored nodes as visualized in figure 5.7. We construct this transformation in the following for the two-dimensional case, a transformation for the three-dimensional case can be constructed analogously.

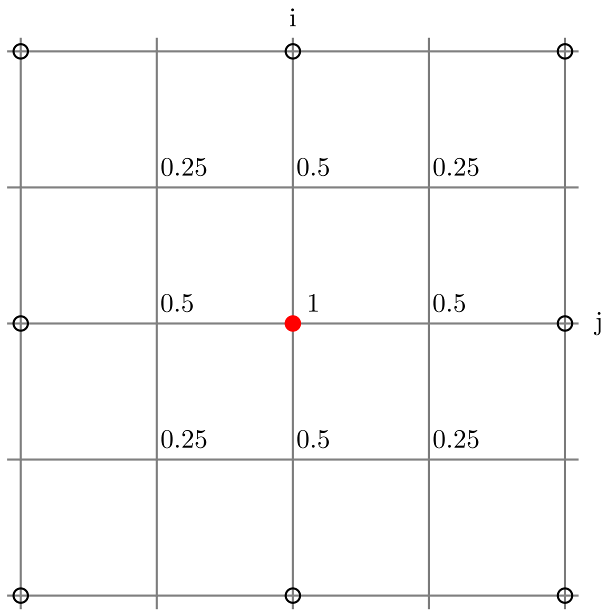


Figure 5.7: The grid displays a fine mesh that shall be transformed in a coarser mesh, where only the marked nodes shall be kept. The value for the red marked node at position (i, j) in the transfer matrix is computed as a linear combination of the value at this point and the values of the surrounding points in the transfer matrix of the fine mesh using the denoted factors. Since the sum of the factors is 4 we have to scale the computed value by $\frac{1}{4}$.

First we want to construct an operator mapping the index of a node in the coarse mesh to the index of the same node in the fine mesh wherefore we use the full meshes. Let us assume we want to reduce a mesh of size $N \times N$ containing $N_f = (N+1)^2$ nodes to a mesh with of size $\frac{N}{2} \times \frac{N}{2}$ containing $N_c = (\frac{N}{2}+1)^2$ nodes. We denote the ratio of the numbers of nodes by $r = \frac{N_f}{N_c}$.

First, we note that if we enumerate the node in a row-major order the $j-th$ node in the coarse mesh is at position

$$p_C(j) = \left(\left\lfloor \frac{1}{\sqrt{N_c}} \right\rfloor, j \bmod (\sqrt{N_c}) \right).$$

It corresponds to the point at position

$$p_F(j) = \left(r \cdot \left\lfloor \frac{1}{\sqrt{N_c}} \right\rfloor, r \cdot (j \bmod (\sqrt{N_c}))\right)$$

in the fine mesh. Therefore, the index of the same node in the fine mesh is given by the following function

$$n(j) = (\sqrt{N_f}) \cdot \left(r \cdot \left\lfloor \frac{1}{\sqrt{N_c}} \right\rfloor\right) + \left(r \cdot (j \bmod (\sqrt{N_c}))\right).$$

Based on these observations we execute the following steps to transform the transfer matrix T_f of the fine mesh.

1. We create a transformation matrix $R \in \mathbb{R}^{N_F \times N_C}$ with entries

$$r_{i,j} = \begin{cases} 1, & \text{if } i = n(j) \\ 0.5, & \text{if } i = n(j) \pm 1 \text{ or } i = n(j) \pm N_F \\ 0.25, & \text{if } i = n(j) \pm (N_F + 1) \pm 1 \\ 0, & \text{else} \end{cases}$$

   $\forall i = 1, ..., N_F, j = 1, ..., N_C$.

2. Since the transfer matrix is defined on the reduced mesh containing only nodes inside the head domain we have to apply the reduction vectors v_f and v_c that contains a 1 for each node of the full mesh that is contained in the reduced mesh. We set $V_f = \text{diag}(v_f)$ and $V_c = \text{diag}(v_c)$ and select the columns and rows corresponding to nodes contained in the reduced meshes by

$$U = V_f R V_c \in \mathbb{R}^{n_f \times n_c}.$$

3. We compute a scaling vector $f \in \mathbb{R}^{n_c}$ with entries

$$f_j = \frac{1}{\sum_{i=1}^{n_f} r_{i,j}}$$

   $\forall j = 1, ..., n_c$. Then we compute

$$W = UF \in \mathbb{R}^{n_f \times n_c}$$

   with $F = \text{diag}(f)$.

4. Finally, we compute the transfer matrix on the coarse mesh by

$$T_c = T_f W \in \mathbb{R}^{S \times n_c}$$

   where $T_f \in \mathbb{R}^{S \times n_f}$ denotes the transfer matrix on the fine mesh.

Hierarchy 3: Source model
Since it has high computational costs we do not use the subtraction approach. We will use Venant and Partial Integration as source models. As discussed before, the Venant approach leads to slightly higher computation costs. Hence, we can use the Partial integration approach on coarse levels to decrease the costs.

Besides constructing the level hierarchies we have to set other parameters to run the MLDA algorithm. The general settings and the initialization of the MLDA algorithm are similar to those of the MH algorithm described in chapter 5.2.

In the following we will construct hierarchies consisting of two or three levels. After constructing a hierarchy of levels, we can define the arguments of the MLDA algorithm. The proposal density q_0 is chosen analogously to the proposal density in the MH algorithm in equation (5.2). The posterior distributions $\pi_0(\cdot), ..., \pi_l(\cdot)$ are also chosen analogously to that from the MH algorithm defined in equation (5.3). Note that they differ at each level since we solve the forward model using different transfer matrices and/or source models.

In addition we have to define a probability mass function to draw the subchain lengths at level i. In the following we use discrete uniform distribution on the interval $[a_i, b_i]$:

$$p_i(x) = \frac{x - a_i + 1}{b_i - a_i + 1} \quad \forall x \in [a_i, b_i].$$

We choose $a_i = 2$ for all levels i, but set b_i independent at each level.

5.4 Implementation

We want to introduce the software used to implement the experiments and give a brief overview of the architecture of our program. Two software components build the basis of the implementation: DUNEuro [50] and MUQ [42]. They communicate with each other using the UM-Bridge framework [55].

DUNEuro
DUNeuro is "an open-source software toolbox for the numerical computation of forward solutions in bioelectromagnetism. Its main focus is to provide an extendable and easy-to-use framework for using various finite element method (FEM) implementations for different neuroscientific applications, such as the electroencephalography (EEG) or magnetoencephalography (MEG) forward problems" ([50]). It is used to solve the forward problem by computing transfer matrices and applying them.

MUQ
"The MIT Uncertainty Quantification library (MUQ) is a modular software framework for defining and solving uncertainty quantification problems involving complex models." ([42]). It includes implementations of various Monte Carlo algorithms, especially the MH and the MLDA algorithm that we want to use.

UM-Bridge
"UM-Bridge (the UQ and Model Bridge) provides a unified interface for numerical models that is accessible from virtually any programming language or framework. It is primarily intended for coupling advanced models (e.g. simulations of complex physical processes) to advanced statistics or optimization methods." ([55])

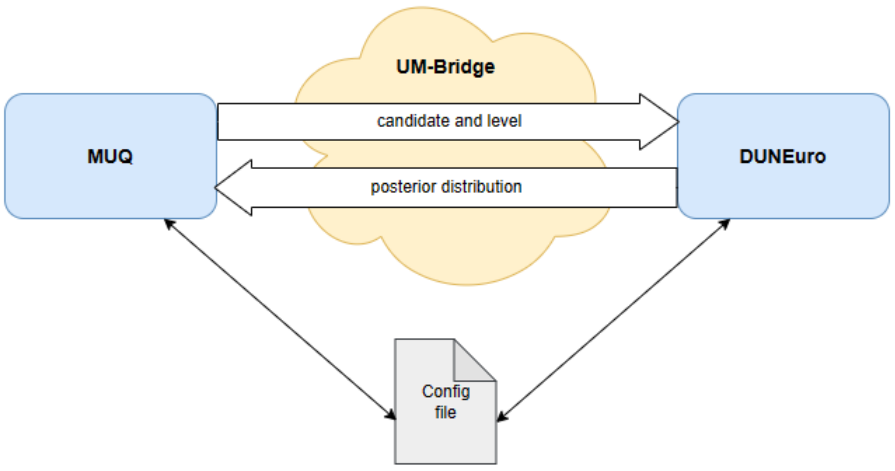


Figure 5.8: Architecture of the program using MUQ and DUNEuro separated from each other. Both are reading from the same configuration file. Communication finds place via UM-Bridge. The mathematical part using MUQ sends a candidate and level information to the model part using DUNEuro which computes the posterior distribution and returns it.

Figure 5.8 displays the architecture of the program. There is one part running the MH and MLDA algorithm using MUQ. If the algorithm has generated a candidate, it sends this candidate and an information about the level it is working on to the other part. This part contains the EEG-model and uses DUNEuro to solve the forward problem and compute the posterior density of the given candidate. It returns this posterior to the first part which can proceed the algorithm. The communication takes place via UM-Bridge, a framework using HTTP. The both separated parts access the same configuration file to ensure consistency.

5.5 Code availability statement

The program code that supports the findings of this thesis is available at `https://zivgitlab.uni-muenster.de/ag-engwer/theses/2022-masterarbeit-wittig`.

6 Results in a two-dimensional setting for simulated data

This chapter pursues two objectives. On the one hand we aim to present the results of the reconstruction of the source distribution and show how the uncertainty of the reconstructed dipole is computed. On the other hand we will discuss the performance of the MH and the MLDA algorithm. All results are generated based on the implementations done in the context of this thesis. The experimental setup is explained in the previous chapter.

Through this chapter we use a two-dimensional setting, since this reduces the run time of the algorithms. Another benefit is, that we are able to view the results in a more intuitive way.

Furthermore, we do not use actually measured potentials but simulated data, generated executing the following initialization steps:

1. We choose a test source $s_{ref} \in [0, 256] \times [0, 256] \times [0, 2\pi)$ (position and orientation). We restrict this source to be positioned in the gray matter. s_{ref} is unknown in practice and thus is not allowed to be used in the following algorithms. It can be used afterwards to compare the results of the algorithms with the correct solution.

2. We use the transfer matrix to compute the expected values $b_{ref} \in \mathbb{R}^M$ at the sensors. In our experiments we choose the subtraction approach as the source model for this step.

3. We determine the variance $\gamma_i^2 \in \mathbb{R}$ for each sensor i. Following [36] we do not set γ_i directly but choose the so called relative noise level $\alpha \in [0, 1]$ The variance is computed as

$$\gamma_i = \alpha \cdot \|b_i\|_{L2},$$

   where $b_i \in \mathbb{R}$ is the measurement value at the i-th electrode in the noiseless case. In the following we set $\alpha = 0.05$.

6.1 Uncertainty quantification

First, we want to show the results of the reconstruction of the source distribution. The results are similar for the MH and the MLDA algorithm because both are converging towards the same distribution (see chapter 4). We use the settings noted in table 6.1 on the finest level.

Parameter	**Value**
Mesh	hexahedral, 256×256
Electrodes	36
Source model	Venant
Relative noise	0.05

Table 6.1: Settings used in the experiments in this chapter.

In the first experiment we set the test dipole $s_{ref} = (120, 222, 1.5)$ (see figure 6.1), simulate measurement values as explained before and sample from the constructed posterior distribution. From the generated samples we can get the three-dimensional empirical distribution of the source as stated in definition 4.9.

The results are displayed in figure 6.3a showing the empirical probability mass function of the source position. Computing the relative number of observations lying in a cell gives the probability of the source to be positioned there. We used the hexahedral mesh as a discretization of the domain. However, displaying a finer resolution of the resulting distribution is possible as well. Further, in each cell with a probability of containing the source higher than 0.01, the mean of the reconstructed orientation is visualized by an arrow. This is a simplification because actually we have a probability distribution in this dimension as well and not just the mean.

In figure 6.3a we can see that the algorithm has reconstructed the dipole correctly. It is contained in the cell with the highest probability to be the source position. Furthermore, we can see, that some other cells are likely to be the

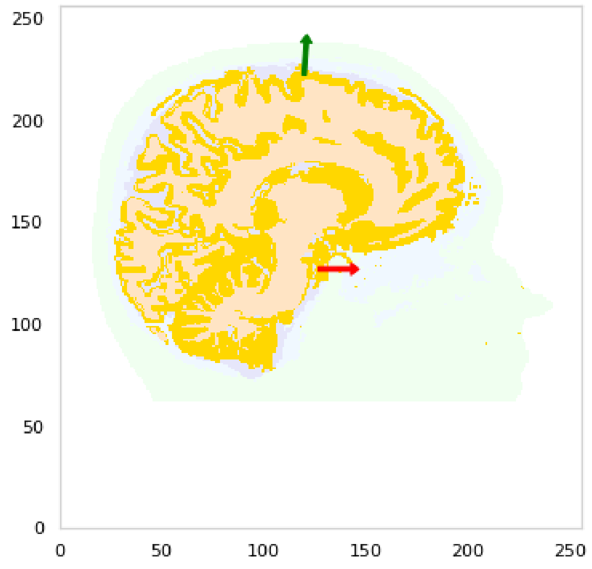


Figure 6.1: Head model with gray matter depicted in yellow. The start dipole at $\theta_0 = (127, 127, 0)$ (red arrow) and the actual dipole that should be reconstructed at $s_{ref} = (120, 222, 1.5)$ (green arrow) are displayed.

source position, too. Additionally, it is visible that positions that lay in the gray matter are preferred by the MH algorithm. These are results we have expected since we constructed the posterior distribution using the gray-matter probability as a factor.

Figure 6.3b shows the dependence of the results on the variance that has been set in the posterior distribution. In our experiments it is given by the relative noise ratio. Increasing the variance of the posterior distribution means assuming a higher uncertainty of the measurement values. Consequently, it results in a higher variance of the reconstructed probability distribution.

Moreover, the reconstructed source distributions of different dipoles are differing in its form and variance even for the same relative noise ratio at the electrodes. They depend on the position and orientation of the actual source. Figure 6.2 shows some reconstructed source distributions to illustrate this assertion. These results show clearly that it is a good approach to quantify the uncertainty of the reconstructed source for each specific case because this includes factors as the eccentricity of the dipole or the head geometry around the dipole position.

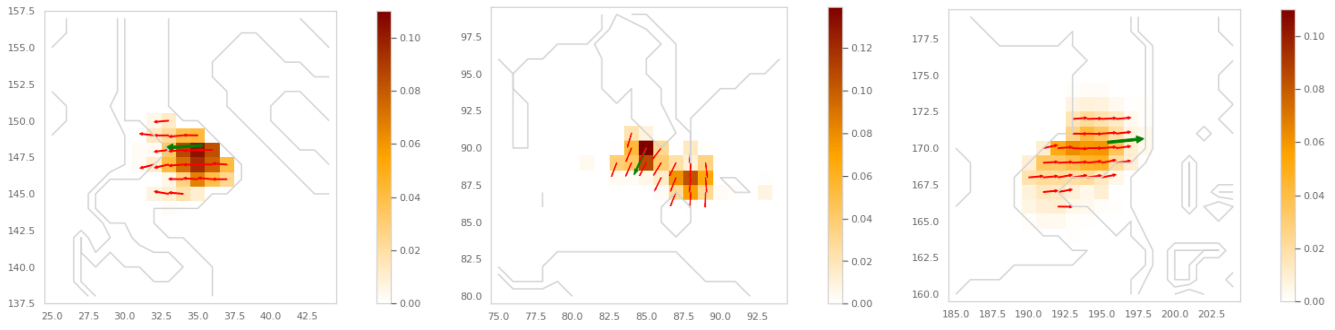
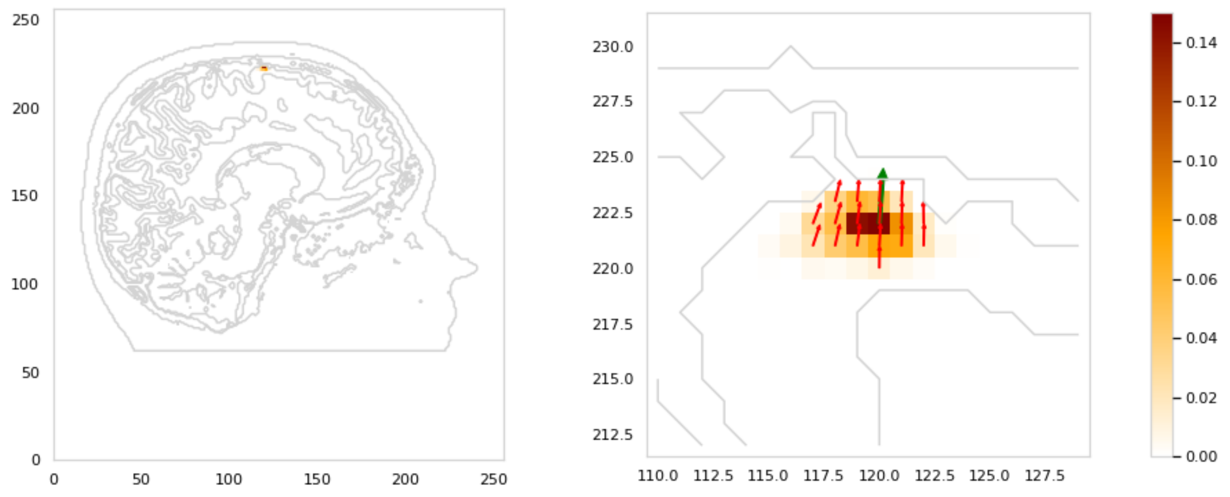


Figure 6.2: Reconstructed probability distributions for different dipoles with the same relative noise ratio.


(a) Generated probability distribution for a relative noise ratio of 5%.

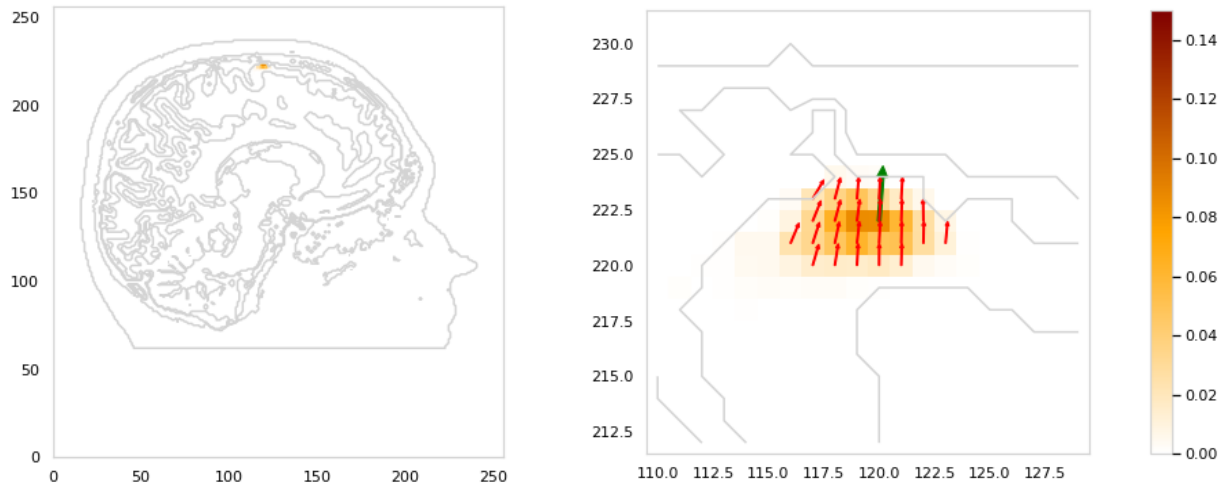

(b) Generated probability distribution for a relative noise ratio of 10%.

Figure 6.3: Probability mass function of reconstructed dipole positions and mean of the orientations reconstructed for each cell. In the images on the left the full domain is shown and the reconstructed area lies in the upper center. In the images on the right an enlarged view of the relevant part is shown. The green arrow indicates the reference dipole.

6.2 Performance

In this chapter we want to analyze the performance of the MH and the MLDA algorithm. We will discuss the choice of the parameters for the MH algorithm. Furthermore, we will evaluate the impact of different hierarchies for the MLDA algorithm on the performance. Finally we will compare the MH and the MLDA algorithm.

Through all evaluations there are two points that have to be considered:

1. Starting at a given point (e.g. the center of the domain or a random point in the domain) the chain has to move close to the dipole position. We call this the burn-in phase. The goal is to decrease the length of this phase.

2. Afterwards, the algorithm has to sample from the posterior distribution. We call this the sampling phase. The goal is to minimize the autocorrelation of the chain generated in this phase.

6.2.1 MH algorithm

In all following experiments we use the settings from table 6.1 to ensure the correct source reconstruction as shown in the previous chapter.

First we observe the reconstruction of one test dipole $s_{ref} = (120, 222, 1.5)$ and initialize the Markov chain with the dipole $\theta_0 = (127, 127, 0)$ (see figure 6.1). We set the proposal variance $(\sigma_x, \sigma_y, \sigma_\rho) = (1, 1, 0.05)$ and run the algorithm for $5 \cdot 10^5$ steps.

To evaluate the quality of the generated sample and justify the choice of parameters, we apply the diagnostic methods discussed in chapter 4.5. Figure 6.4 shows the trace plots for the samples generated by the MH algorithm. These results show that a burn-in phase of at least $2 \cdot 10^3$ steps is recommendable.

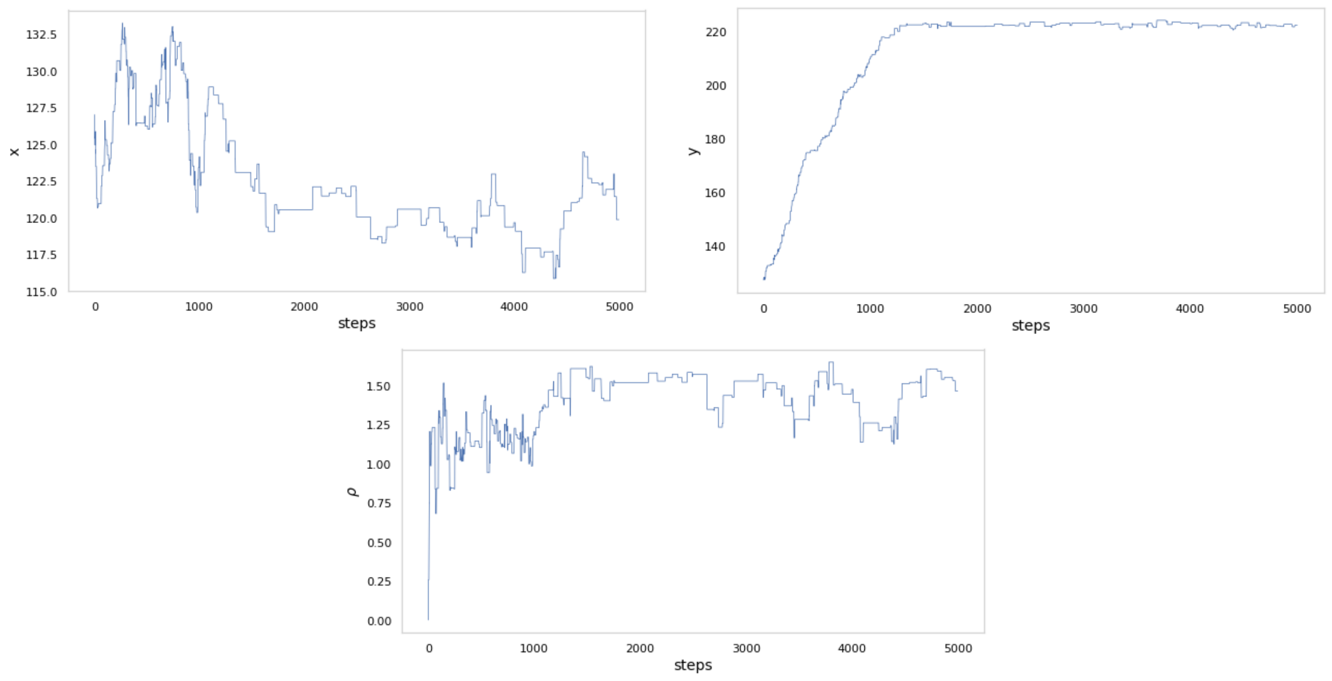


Figure 6.4: Trace plot of the first 5000 samples of the three dimensions of the generated Markov chain starting at $(127, 127, 0)$ moving to the dipole $(120, 222, 1.5)$. The burn-in and the sampling phase are clearly visible. The plots show the convergence of the chain and that a burn-in phase of 2000 samples is recommendable.

Regarding the efficiency of the algorithm in the sampling phase, the trace plots in figure 6.4 give a first impression of a relatively high autocorrelation of samples, since the chain seems to sample from the same region for some steps before moving away. A more reliable assertion can be made considering the autocorrelation plots in figure 6.5. They show that the autocorrelation behaves differently in each dimension. It is decreasing exponentially for the $y-$coordinate of the position which is the desired behaviour. For the orientation angle ρ the decrease is also exponential but with a lower rate. The autocorrelation of the $x-$coordinate is decreasing rather linearly - there is a high correlation between samples of the chain lying many steps apart.

These observations are justified by computing the effective sample size and the integrated autocorrelation time of the generated Markov chain which are $\hat{N}_{eff} = 184$ and $\hat{\tau}(x) = 545$. That means we have to run the chain for 545 steps to generate an observation that is independent of the initial one.

Note that a general classification of the decrease of the autocorrelation as “fast” or “slow” is very unspecific since the characterization of a “good” autocorrelation is highly dependent on the problem setting. It makes more sense comparing autocorrelations of the same algorithm with different parameters. Further, different algorithms as the MH and the MLDA algorithm can be compared.

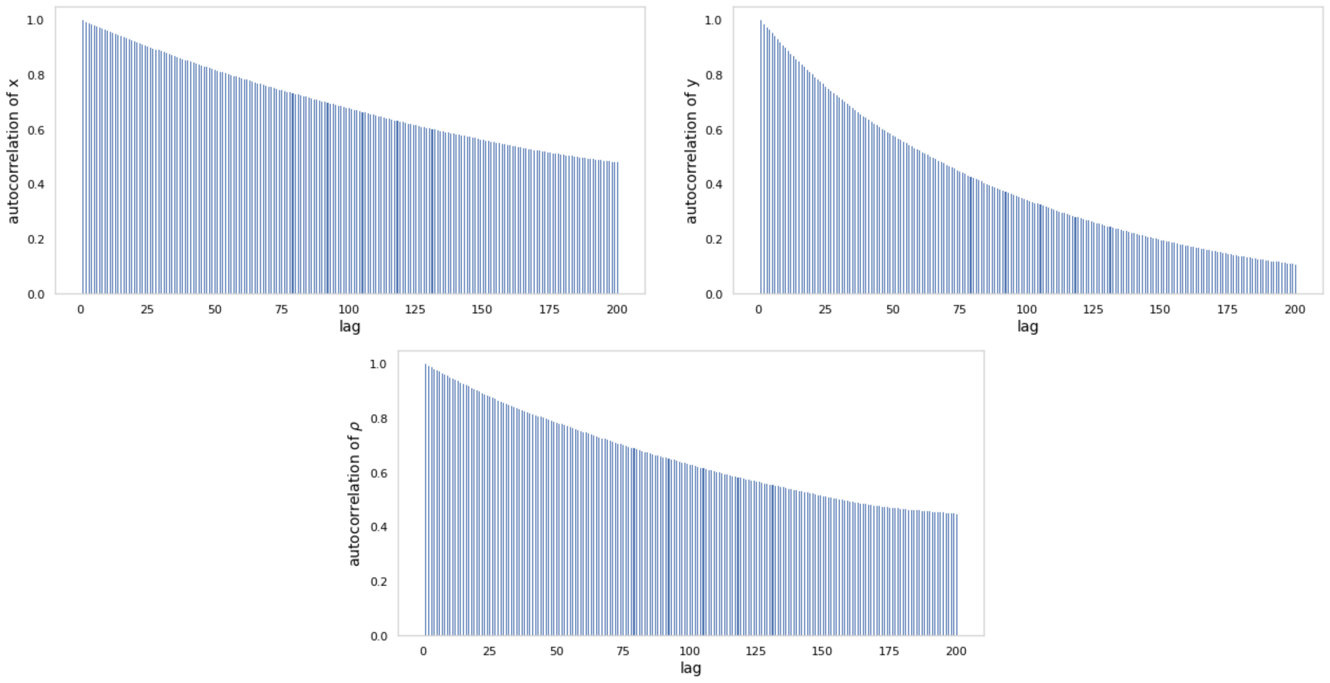


Figure 6.5: Autocorrelation plots of the three dimensions of the Markov chain generated by the MH algorithm. They show that the autocorrelation is decreasing with a different rate in each dimension.

The chosen proposal variance affects the performance of the MH algorithm strongly. However, there is no parameter choice that is perfect for all dipoles. Therefore, we search a proposal variance that leads to the best results on average. For this purpose, we run the algorithm with a certain proposal variance multiple times reconstructing a number of different dipoles and compute the integrated autocorrelation time (IAT) of each trial. Since the autocorrelation is computed dimension-wise (see chapter 4.5) we use the highest IAT, i.e. $\hat{\tau} = \max(\hat{\tau}_x, \hat{\tau}_y, \hat{\tau}_\rho)$. Then we can compute the mean, the median and the variance of the IAT and compare these values between different algorithm configurations. Table 6.2 shows the results of these investigations and the boxplot in figure 6.6 provides a visualization.

Label	**Proposal variance**	**Burn-in**	**Integrated autocorrelation time**		
	$(\sigma_x, \sigma_y, \sigma_\rho)$		**Mean**	**Median**	**Variance**
MH 1	$(0.5, 0.5, 0.025)$	$50,000$ samples	3,853	460	34,529,305
MH 2	$(1, 1, 0.05)$	$50,000$ samples	750	283	733,627
MH 3	$(2, 2, 0.1)$	$30,000$ samples	1,683	421	12,276,671

Table 6.2: Comparison of the performance of the MH algorithm for different proposal variances.

Considering the IAT, a proposal variance of $(\sigma_x, \sigma_y, \sigma_\rho) = (1, 1, 0.05)$ is clearly the preferable choice since it results in the lowest mean, a low median and the lowest variance of the IAT.

Regarding the burn-in phase, the results show that in the first two experiments a burn-in of $50,000$ samples is recommendable. In that time the generated chain has converged towards the target distribution in each experiment. The burn-in size decreases as we increase the proposal variance. Considering the run time of the algorithm the settings of experiment MH 2 are still superior. This can be shown by a simple calculation. Let us assume we want an effective sample size of 10^4 samples. In algorithm MH 2 we need to run the chain for

$$N = 5 \cdot 10^4 + 750 \cdot 10^4 = 7.55 \cdot 10^6$$

steps. Using algorithm MH 3 we need

$$N = 3 \cdot 10^4 + 1683 \cdot 10^4 = 1.686 \cdot 10^7$$

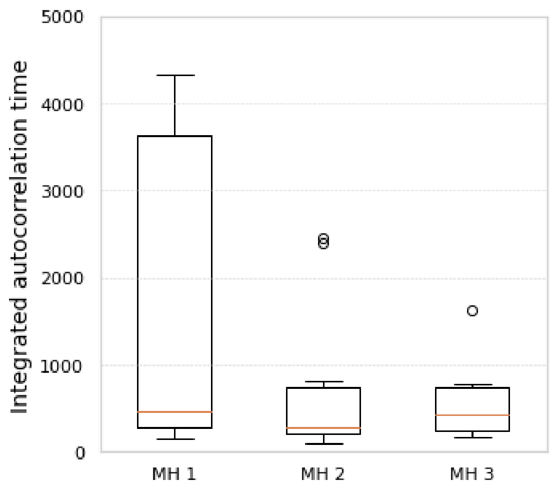


Figure 6.6: Integrated autocorrelation time of the MH algorithm with varying step size of the proposal starting with a small step size on the left and increasing it towards the right. The full results can be found in table 6.2.

steps.

6.2.2 MLDA algorithm

In this section we want to analyze and compare the performance of different MLDA algorithms, in particular the different hierarchies. An analysis of the costs as well as a comparison with the MH algorithm will be subject of the next section. The full parameter choices of the MLDA algorithms we use can be found in appendix A. They have been optimized through multiple trials with the goal of minimizing the mean of the IAT.

We construct MLDA algorithms consisting of two levels. The settings on the fine level are the same as in the MH algorithm in the last section (see table 6.1). In chapter 5.3 we discussed different hierarchies regarding the mesh width, the number of electrodes and the source model. First we analyze each of them separately. The results are presented in table 6.3 and figure 6.7.

Label	**Hierarchy**	**Subchain length (mean)**	**Burn-in**	**Integrated autocorrelation time**		
				Mean	**Median**	**Variance**
	Mesh					
MLDA M1	128×128	11	7,000	363	192	145,354
MLDA M2	64×64	11	2,000	566	399	186,036
	Electrodes					
MLDA E1	18	11	5,000	84	41	13,572
MLDA E2	9	11	5,000	77	35	7,614
MLDA E3	5	11	6,000	177	53	63,784
	Source Model					
MLDA S1	Partial Integration	11	7,000	138	71	27,761

Table 6.3: Performance results of 2-level MLDA algorithms for different hierarchies.

The outcome of the experiments shows, that especially the electrodes hierarchy is very promising. Running the MLDA algorithm with a reduced number of 18 electrodes on the coarse level leads to a low IAT of 84 on average. That means, after 84 steps of the MLDA algorithm we have generated a sample that is not correlated with the initial one. Another interesting point to consider is the small variance of the resulting IAT when the algorithm is applied for different dipoles, indicating that the algorithm is stable. A number of only 9 electrodes on the coarse level leads to almost similar results. As a matter of fact, the variance is considerable smaller. These results correspond to the considerations in chapter 5.3 where we showed that a high portion of the variance of measurements can be kept by a few uncorrelated sensors. Though, a further reduction to 5 electrodes results in an increase of the IAT.

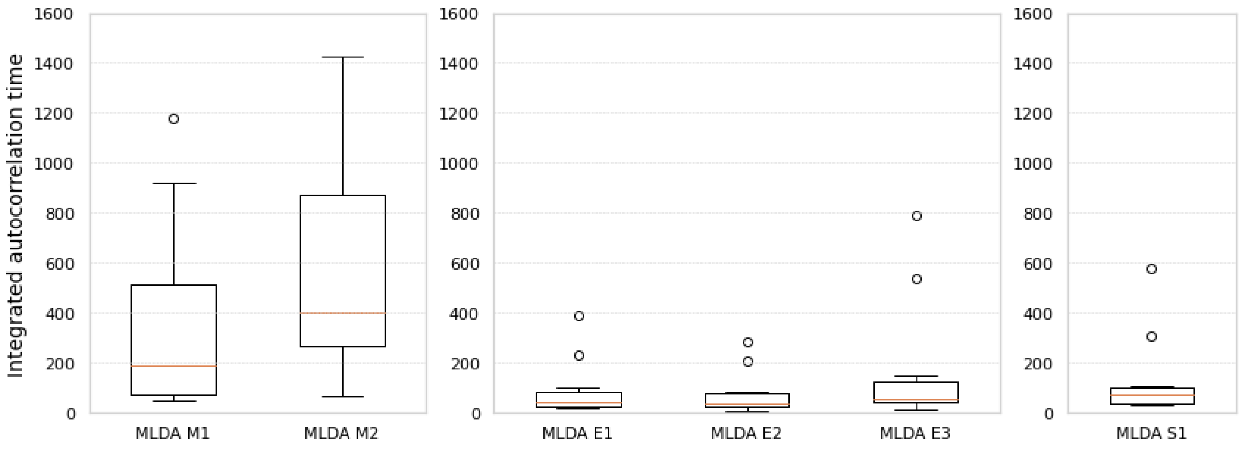


Figure 6.7: Boxplot of the IAT of 2-level MLDA algorithms for reductions concerning the mesh size (left), the number of sensors (middle) and the source model (right) as they are denoted in table 6.3. The results show that the usage of the Partial Integration source model and the reduction of the number of electrodes result in lower autocorrelation times than the reduction of the mesh-size.

The choice of Partial Integration as the source model on the coarse level of the MLDA algorithm generates good results, too. It leads to a stable algorithm with an average IAT of 138.

Regarding a reduction of the number of cells of the used mesh, we observe a massive increase of the IAT. Thus the benefits of such a reduction have to be evaluated carefully. The better choices seem to be the other used hierarchies, but a final decision depends on the concrete setting.

The results for the necessary burn-in time show that the chain converges towards the target distribution in a number of steps of the same size. In each experiment it was possible to reconstruct all dipoles starting at the center of the domain.

Next, we want to combine different hierarchies to evaluate if we can benefit from the cost reductions of different hierarchies. A variety of possibilities exists. In table 6.4 and figure 6.8 we present three MLDA settings that worked well in the tests. In particular a combination of a reduction of sensors and a switch to the Partial Integration source model resulted in a good performance of the algorithm.

Label	**Reductions on coarse level**	**Subchain length (mean)**	**Integrated autocorrelation time**		
			Mean	**Median**	**Variance**
MLDA C1	5 electrodes Partial Integration	11	179	60	40,633
MLDA C1	9 electrodes Partial Integration 128×128 mesh	11	250	78	92,373
MLDA C2	9 electrodes Partial Integration 64×64 mesh	13	255	135	104,983

Table 6.4: Performance results of 2-level MLDA algorithms for combined hierarchies.

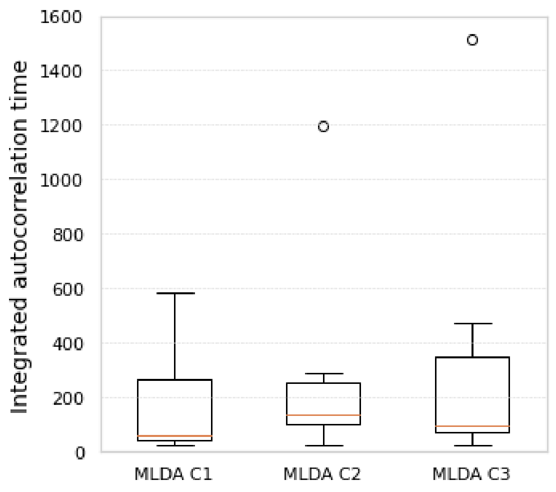


Figure 6.8: Boxplot of the IAT of 2-level MLDA algorithms for combined reductions on the coarse level as they are denoted in table 6.4. The results show that the combination of the reduction of sensors and a switch of the source model from the Venant approach to Partial Integration results in low autocorrelation times. An additional combination with the reduction of the cell size in the used mesh is possible but increases the autocorrelation times noticeable. An additional combination with the reduction of the cell size in the used mesh is possible but increases the autocorrelation times noticeably.

6.2.3 Comparison of MH and MLDA

To compare the different algorithms we want to consider the computational costs. As we have already discussed in section 5.3, the determination of costs is not straightforward since it depends on the implementation and the hardware. Based on theoretical considerations, we stated that the major part of the costs arises in the computation of the matrix-vector multiplication in the step of solving the forward problem. This multiplication has to be executed once per generated sample. On the other hand, we found that there are fixed costs of not negligible size dependent on the implementation.

Therefore, we divide the total costs in two parts, a fixed and a variable one,

$$c = c^f + c^v. \tag{6.1}$$

c^f denotes the fixed part including all costs that arises independent of the mesh size, the number of sensors and the chosen source model. The variable part is denoted by c^v and is dominated by the computational costs of the matrix-vector product.

For level l we denote by n_l the mean of the probability distribution the subchain length is drawn from. The costs of running one step of the MH algorithm at this level are denoted by c_l. Then, the total costs for drawing one sample with the MLDA algorithm consisting of L levels are

$$c = \sum_{l=0}^{L} \left(\prod_{k=0}^{l} n_l \right) \cdot c_l. \tag{6.2}$$

Following equation (6.1) we denote the costs per sample on the fine level L as

$$c_L = c^f + c_L^v.$$

Then the costs on the coarse level l are given by

$$c_l = c^f + c_l^v = c^f + \alpha_l c_L^v. \tag{6.3}$$

The factor α_l has to be determined in each specific setting. It is the factor of the reduction of variable costs between level L and level l.

Using equations (6.2) and (6.3) the total costs of the MLDA algorithm can be computed as follows:

$$\begin{aligned} c &= \sum_{l=0}^{L} \left(\prod_{k=0}^{l} n_l \right) \cdot c_l \\ &= \sum_{l=0}^{L} \left(\prod_{k=0}^{l} n_l \right) \cdot (c^f + \alpha_l c_L^v) \\ &= \left(\sum_{l=0}^{L} \left(\prod_{k=0}^{l} n_l \right) \right) \cdot c^f + \left(\sum_{l=0}^{L} \alpha_l \left(\prod_{k=0}^{l} n_l \right) \right) \cdot c_L^v \end{aligned}$$

In the 2-level case this reduces to

$$c = (1+n) \cdot c^f + (1 + \alpha \cdot n) \cdot c_{fine}^v \tag{6.4}$$

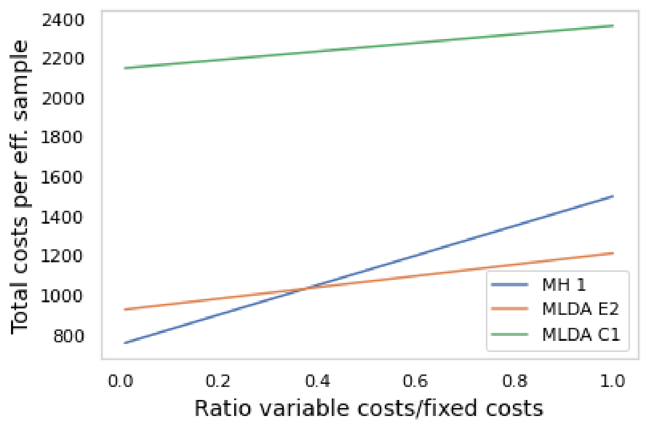
To compare the MH and the MLDA algorithms presented in the previous two sections, we compare the costs per effective sample, which we compute by multiplying the IAT with the costs per sample. The reduction factor α is computed using the test program we already used in section 5.3 to estimate the run time of the matrix-vector multiplication. The results are shown in table 6.5.

Algorithm	**Autocorrelation**	α	**n**	**costs per sample**	**costs per effective sample**
MH 1	750	-	-	$c^f + c_{fine}^v$	$750c^f + 750c_{fine}^v$
MLDA M1	363	0.6875	11	$12c^f + 8.5625c_{fine}^v$	$4356c^f + 3108c_{fine}^v$
MLDA M2	566	0.25	11	$12c^f + 3.75c_{fine}^v$	$6792c^f + 2.122c_{fine}^v$
MLDA E1	94	0.5	11	$12c^f + 6.5c_{fine}^v$	$1128c^f + 611c_{fine}^v$
MLDA E2	77	0.25	11	$12c^f + 3.75c_{fine}^v$	$924c^f + 288c_{fine}^v$
MLDA E3	177	0.14	11	$12c^f + 2.5c_{fine}^v$	$2124c^f + 443c_{fine}^v$
MLDA S1	138	0.25	11	$12c^f + 2.4c_{fine}^v$	$1656c^f + 331c_{fine}^v$
MLDA C1	179	0.01875	11	$12c^f + 1.20625c_{fine}^v$	$2148c^f + 216c_{fine}^v$
MLDA C2	250	0.01875	11	$12c^f + 1.20625c_{fine}^v$	$3000c^f + 301c_{fine}^v$
MLDA C3	255	0.0125	13	$14c^f + 1.1625c_{fine}^v$	$3570c^f + 296c_{fine}^v$

Table 6.5: Computational costs of the MH and the MLDA algorithm for different parameter settings computed using equation (6.4). The reduction factor α is determined by comparing the run time of the multiplication of the dense transfer matrix with the sparse right hand side vector for the algorithm parameters. There is no algorithm with the absolute lowest costs. Depending of the rate of fixed and variable costs one of the blue marked algorithms should be chosen.

These results show that the choice of the most efficient algorithm depends on the ratio of the fixed and variable costs. Out of the presented algorithms there are three of them outperforming the others: the MH algorithm itself (MH 1), the algorithm quartering the number of electrodes (MLDA E2) and the algorithm combining a sharp reduction of the number of electrodes with the switch to Partial Integration as source model (MLDA C1).
Figure 6.9 shows the dependency of the total costs on the ratio of variable and fixed costs. It is visible that the MH algorithm is only preferable for fixed costs more than twice the variable costs. Afterwards the total costs increase rapidly. If the ratio lies between 0.4 and 17, the most efficient algorithm is the MLDA E2 algorithm reducing only the number of electrodes. For variable costs that are more than 17-times of the fixed costs the MLDA C1 algorithm, combining a reduction of sensors and the Partial Integration source model is superior.

In summary, the experiments have shown that a properly constructed MLDA algorithm comes with the benefit of being stable (low variance of IAT) and having low total costs per effective sample, provided the implementation is efficient and the overhead is kept small.



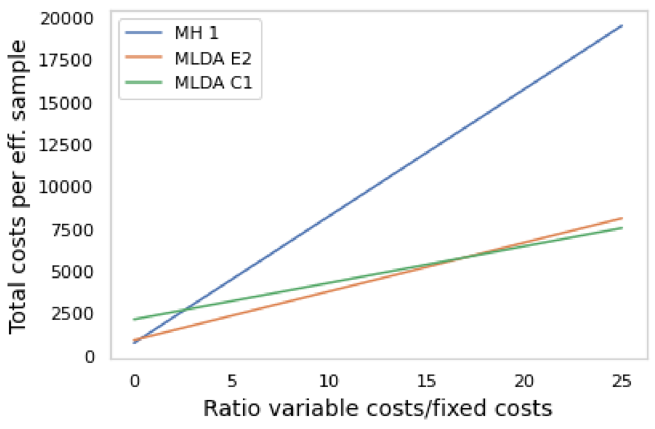


Figure 6.9: Dependency of the total costs on the ratio of variable and fixed costs. The y-axis indicates the total costs per effective sample as a factor, that has to be multiplied with the value of the fixed costs. The image on the left shows the total costs for a ratio smaller than 1, the image on the right displays fixed costs higher than variable costs.

7 Results in a three-dimensional setting for measured data

So far, we applied and analyzed the algorithms in a two-dimensional setting with simulated data. In this chapter we want to apply the MH and the MLDA algorithms to real data. Consequently, we work in a three-dimensional setting. The EEG data we use has been measured in a somatosensory experiment, stimulating the median nerve at the right wrist of the subject. We will use this data to show, that our algorithms indeed work not only for simulated but also for real data. Furthermore, we will justify several theoretical considerations we made in the previous chapters, concerning for example the correlation of the sensors.

7.1 Data analysis

The available data includes measurements of $N = 2105$ trials, each consisting of measurements from 50ms before the stimulus until 150ms after the stimulus. Viewing the butterfly plot in figure 7.1 we observe different signal components. We will try to reconstruct the negative peak occurring 24ms after the stimulus. We analyze the values measured at this point. As a result, the time dimension is not part of the further analysis and we have a data matrix $D = (d_{i,j})_{i,j} \in \mathbb{R}^{N\times M}$ where $M = 69$ denotes the number of sensors.

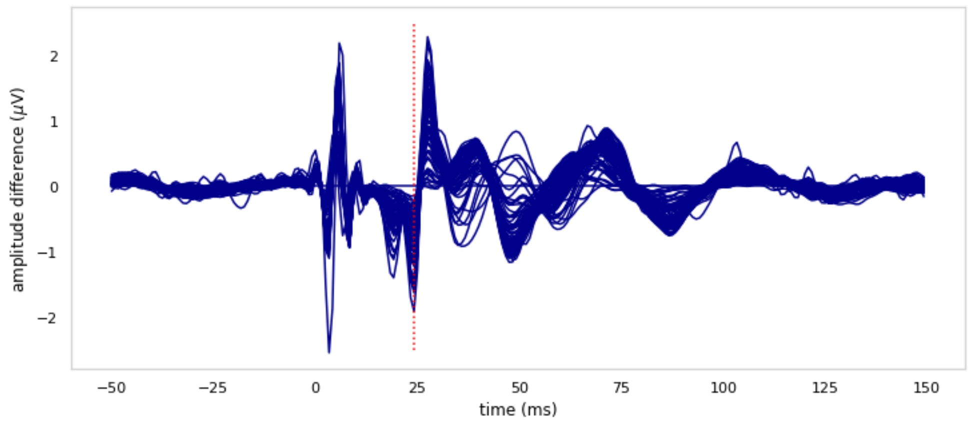


Figure 7.1: Butterfly plot showing the amplitude differences for each of the 69 channels. The red line indicates the point at which we want to reconstruct the source of the signal.

As discussed in the previous chapter, we first have to unify the data by centering the measured sensor values for each trial. For each row we compute the mean of the measurement values and subtract it. We get $D^c = (d^c_{i,j})_{i,j} \in \mathbb{R}^{N\times M}$ with entries

$$d^c_{i,j} = d_{i,j} - \frac{1}{M}\sum_{k=1}^{M} d_{i,k}.$$

In the next step, we compute the sample mean of the measured sensor values over all trials, which is called an evoked potential

$$\bar{X} = \frac{1}{N}\sum_{i=1}^{N} d_i \in \mathbb{R}^M$$

where d_i is the $i-$th row of D.

Since we only reconstruct position and orientation but not the strength of the dipole, we have to normalize the measured values. We normalize using the L2-Norm of the sample mean and get

$$b_{ref} = \frac{1}{\|\bar{X}\|_{L2}}\bar{X} \in \mathbb{R}^M.$$

This is the vector of measurements we will use to reconstruct the source. A visualization can be seen in figure 7.2. Furthermore we get the normalized centered data matrix

$$D^n = \frac{1}{\|\bar{X}\|_{L2}} D^c \in \mathbb{R}^{N\times M}.$$

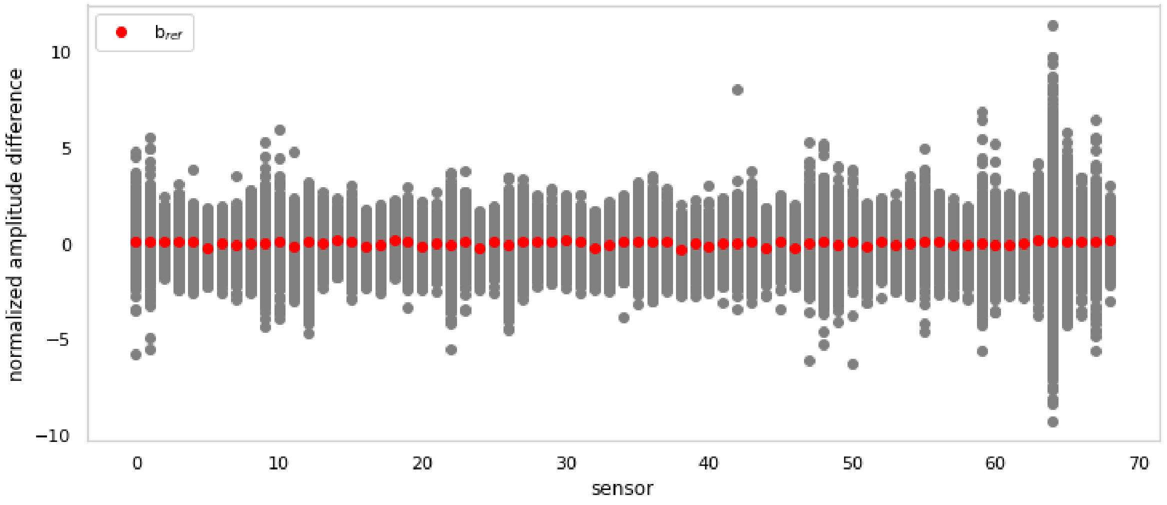


Figure 7.2: The gray values show the measured potential differences at the electrodes in each of the 2105 trials. The red points are the average values of all trials. The figure shows clearly that the measured values vary strongly. Consequently, building the average of a high number of trials is crucial.

In the next step we want to estimate the variance of the measured values at the sensors. As can be seen in figure 7.2, the variance between the measured values of single trials is huge and averaging is a standard procedure. Therefore, we are interested in the standard error of the mean. For sensor j it is given by

$$SEM_j = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} \sqrt{\left(\frac{(d_{i,j}^n - b_j)^2}{N-1} \right)} = \sqrt{\frac{1}{N} Var(d_{1,j}^n, ..., d_{N,j}^n)}$$

(see [21]). We use the square of the SEM as variance for our posterior distribution:

$$Var_j = \frac{1}{N} Var(d_{1,j}^n, ..., d_{N,j}^n) \tag{7.1}$$

To verify this choice we estimated the variance from the given data executing the following steps:

1. Draw $\lfloor N/k \rfloor$ disjoint chunks containing the data of $k < N$ trials.
2. Compute the mean of each chunk.
3. Compute the variance of the chunk means.

The results can be seen in figure 7.3 and show clearly that formula (7.1) holds true.

Next, we compute the sample covariance matrix

$$Q = \frac{1}{N-1} \sum_{i=1}^{N} (d_j - \bar{X})(d_i - \bar{X})^T \in \mathbb{R}^{M \times M}$$

where d_i is the $i-$th row of D. It contains the variances of the sensors in the diagonal. $q_{j,j}$ is the variance of sensor j. We use these variances for the posterior distribution of the MCMC algorithms.

Furthermore we can compute the sample correlation matrix, which is a normalized version of the sample covariance matrix defined as

$$R = VQV$$

where V is a diagonal matrix with diagonal entries $v_{i,i} = \sqrt{\frac{1}{q_{i,i}}}$.

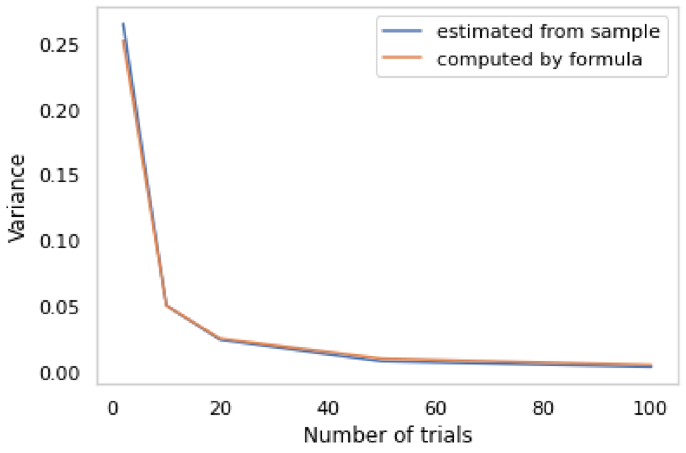


Figure 7.3: Computation of the variance of the mean of a number of trials estimated from the sample by computing the variance of the means of chunks of a certain number of trials. Both values are very similar.

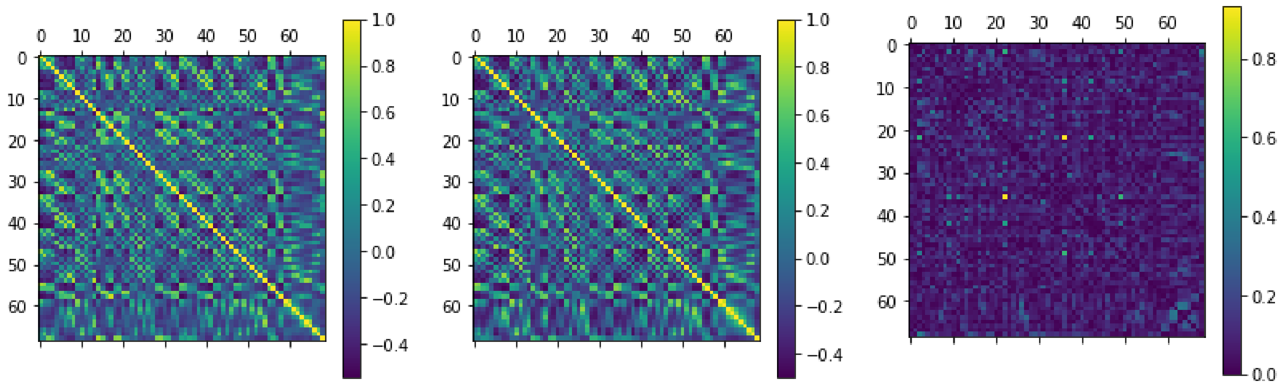


Figure 7.4: Correlation matrix generated from the data (left) and the normalized outer product of the transfer matrix (center) and absolute difference of them (right).

We stated before that the normalized outer product of the transfer matrix provides the correlation of the sensors (chapter 5.3) and want to verify this assumption using the data from the experiment. Figure 7.4 compares the sample correlation matrix generated from the data and the normalized outer product of the transfer matrix, showing that they are indeed very similar.
Mathematically, this can be reasoned by the following argumentation. We modeled the EEG problem using the Poisson equation (see chapter 2) as

$$-\Delta u = f.$$

We assume to have erroneous measurements, thus we get in the $i-$th trial

$$-\Delta u^i = f + \tilde{f}^i$$

where $\tilde{f}^i$ describes the error of the right hand side for trials $i = 1, ..., N.$. To solve the equation we apply the transfer matrix T and get

$$u^i = T(f + \tilde{f}^i).$$

Thus, the mean of the potential is given by

$$\begin{aligned}
E(u) &= \frac{1}{N}\sum_{i=1}^{N} u^i \\
&= \frac{1}{N}\sum_{i=1}^{N} T(f+\tilde{f}^i) \\
&= \frac{1}{N}\sum_{i=1}^{N} Tf + \frac{1}{N}\sum_{i=1}^{N} T\tilde{f}^i \\
&= Tf + T\underbrace{\left(\frac{1}{N}\sum_{i=1}^{N}\tilde{f}^i\right)}_{=0} \\
&= Tf,
\end{aligned}$$

where we use that the average of the errors is equal to 0 and the fact that T is a linear operator. The covariance of the potential at the sensors is then given by

$$\begin{aligned}
Cov(u) &= \frac{1}{N}\sum_{i=1}^{N}(E(u)-u^i)(E(u)-u^i)^T \\
&= \frac{1}{N}\sum_{i=1}^{N}\left(Tf - T(f+\tilde{f}^i)\right)\left(Tf - T(f+\tilde{f}^i)\right)^T \\
&= TT^T\frac{1}{N}\sum_{i=1}^{N}\tilde{f}^i\tilde{f}^{i^T}.
\end{aligned}$$

To get the correlation matrix, the covariance matrix has to be normalized in the sense that each row is multiplied with the inverse of the diagonal entry. Provided a sufficiently large number of trials, the correlation matrix computed from the data is similar to the normalized outer product of the transfer matrix as figure 7.4 shows.

Next, we analyze the eigenvalues of the correlation matrix (see figure 7.5). The comparison between the eigenvalues of the normalized outer product of the transfer matrix and the correlation matrix determined from the data shows almost no difference.

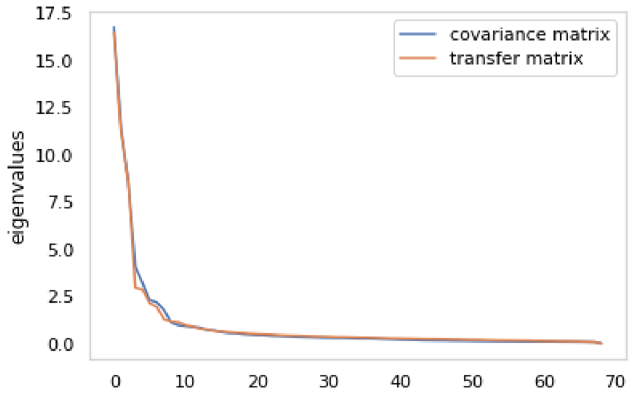


Figure 7.5: Ordered eigenvalues the normalized outer product of the transfer matrix and of the covariance matrix of the data showing that both behave very similarly. For both matrices the eigenvalues decrease fast, which indicates a possible dimension reduction with a small loss of information.

7.2 Source reconstruction

In this section we finally want to apply the MH and the MLDA algorithm to the measured data and reconstruct a source distribution. We run the MH algorithm using the parameters denoted in table 7.1. In addition, we constructed a MLDA algorithm consisting of two levels, reducing the number of electrodes on the coarse level in combination with setting Partial Integration as the source model. These reductions have led to good results in the two-dimensional experiments (see section 6). The full parameter choices can be found in table 7.2. In both algorithms we start at the center of the domain $\theta^0 = (127, 127, 127, 0, 0)$.

Parameter	**Value**
Mesh	hexahedral, $256 \times 256 \times 256$
Electrodes	69
Source model	Venant
Proposal variance	(0.5,0.5,0.5,0.01,0.01)

Table 7.1: Settings used in the MH algorithm applied in this chapter.

Parameter	**Coarse level**	**Fine level**
Mesh	hexahedral, $256 \times 256 \times 256$	hexahedral, $256 \times 256 \times 256$
Electrodes	35	69
Source model	Partial Integration	Venant
Proposal variance	(0.5,0.5,0.5,0.01,0.01)	-
Subchain length	Uniform distributed on $[2, 10]$	-
Variance factor	2	-

Table 7.2: Settings used in the MLDA algorithm applied in this chapter.

Since both algorithms reconstruct the same posterior distribution, the results are equivalent. Therefore, we speak in the following of the MCMC results. The source has been located in the left rear part of the gray matter. We have visualized the results of both algorithms in different figures. The trace plots indicating the performance of the algorithms can be found in appendix B. In figure 7.7a the reconstructed area is shown. In the three-dimensional views the different probabilities can not be seen well because the outermost cells have the lowest probability. For that reason, figure 7.6 shows the reconstructed source distribution in different slices of the brain. We used those slices, containing the cell with the highest probability to be the location of the dipole, which is the cell with the center at $(94.5, 89.5, 188.5)$.

We compare our results with the results generated by a classical method - the dipole scan. Basically, the dipole scan consists of the computation of the pseudoinverse for the local leadfield of a number of possible dipole positions. We choose the centers of the cells lying in the gray matter fulfilling the Venant condition (that is all nodes of this cell belong only to cells of the gray matter). We use the formulas from [40]. For each of those positions p, the local leadfield $L \in \mathbb{R}^{69\times 3}$ is computed for the three moments $(1, 0, 0), (0, 1, 0)$ and $(0, 0, 1)$ and its pseudoinverse L^+ is determined. Then we can compute the relative residual variance

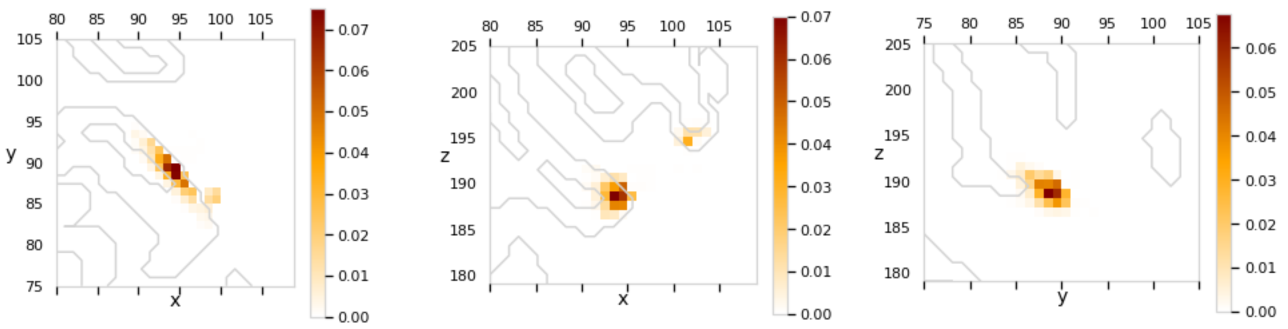
$$rrv(p) = \frac{\|b_{ref} - LL^+ b_{ref}\|_{L2}^2}{\|b_{ref}\|_{L2}^2}$$

with averaged measurement data vector b_{ref} . The goodness of fit (gof) is then computed by

$$gof(p) = 1 - rrv(p).$$

The dipole scan identified the dipole with position $(94.5, 89.5, 188.5)$ and moment $(0.700, 0.557, 0.442)$ as the most likely source of the measured values.

The MCMC algorithms assigns the highest probability to the cell with the center at $(94.5, 89.5, 188.5)$ and reconstructs $(0.627, 0.620, 0.471)$ as the most likely moment in this cell. This shows that both algorithms have similar


(a) Results of MCMC algorithms showing the computed probability of each cell to be the location of the source.

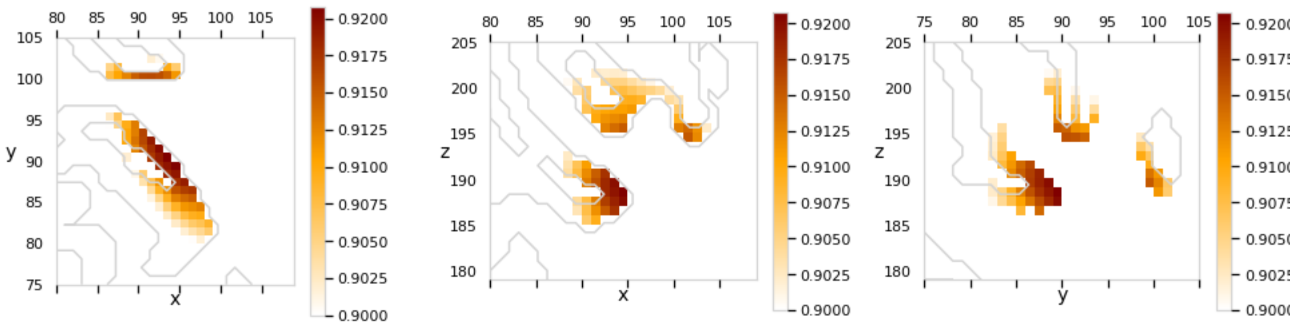

(b) Results of dipole scan picturing all cells with a gof larger than 0.9. The values represent the gof.

Figure 7.6: Source distribution reconstructed by the MCMC algorithms displayed in an axial slice at $z = 188$ (left), a coronal slice at $y = 89$ and a sagittal slice at $x = 94$ (right). The pictures show that both methods assign the highest probabilities to the same cells, but the MCMC algorithms are more specific than the dipole scan.

results regarding the most likely dipole position and moment.

For a further comparison, in figure 7.7b we show the dipole positions detected by the dipole scan with a goodness of fit higher than 0.9 in comparison with the cells found by the MCMC algorithms.

Regarding the computational costs we made the following observations. While for the MH algorithm an IAT of 3.672 samples holds, we get an IAT of only 455 samples for the MLDA algorithm. We explained in chapter 6 how the computational costs can be calculated. Using equation (6.4) we get the following computational costs to generate one sample for the MLDA algorithm in our experiment:

$$\begin{aligned} c &= (1+n)\cdot c^f + (1+\alpha\cdot n)\cdot c^v_{fine} \\ &= 7c^f + (1+6\alpha)\cdot c^v_{fine}. \end{aligned}$$

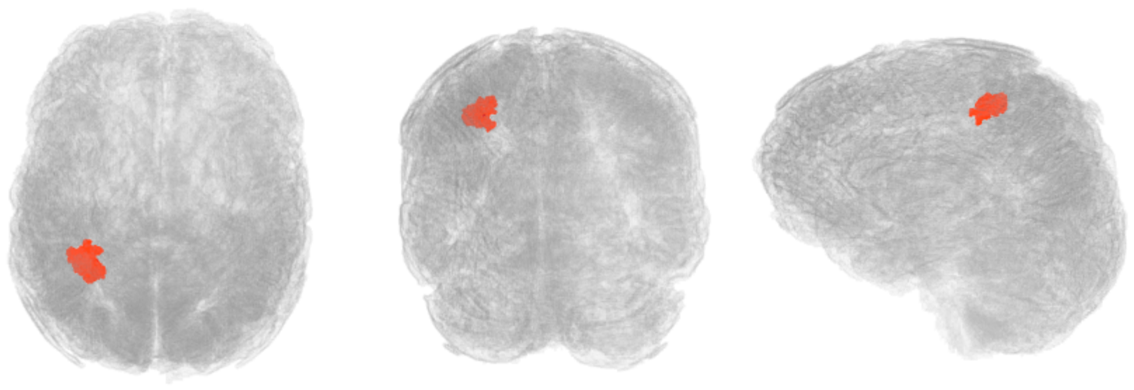
We determine the costs per effective sample by multiplying the costs per sample with the integrated autocorrelation time and get for the MH algorithm

$$c^{MH}_{eff} = 3672c^f + 3672c^v_{fine}.$$

The costs per effective sample for the MLDA algorithm are

$$\begin{aligned} c^{MLDA}_{eff} &= 455\cdot(7c^f + (1+6\alpha)\cdot c^v_{fine}) \\ &= 3185c^f + 455(1+6\alpha)c^v_{fine} \\ &\le 3185c^f + 3185c^v_{fine}. \end{aligned}$$

In the last step we used $\alpha \le 1$, which corresponds to the assumption that the costs on the coarse level are lower than those on the fine level. This assumption is indeed true because we only made reductions from the fine to the coarse level.


(a) Results of the source reconstruction by the MCMC algorithms.

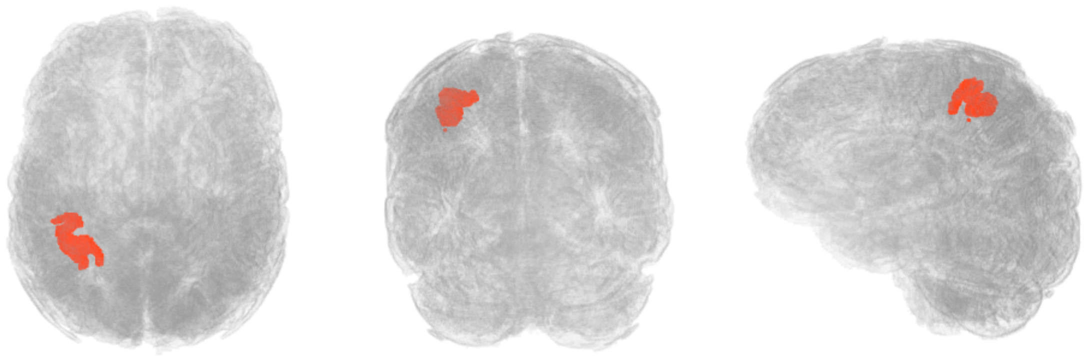

(b) Results of the source reconstruction by the dipole scan technique.

Figure 7.7: Comparison of the results of the MCMC algorithms and the dipole scan. The gray matter containing the reconstructed source distribution is displayed from the top (left), from the back (center) and from the left side (right). Both algorithms located the dipole in the same area.

As a result, in our experiment the MLDA algorithm has been producing less computational costs than the MH algorithm independent of the concrete cost reductions on the coarse level.

A summary of these results can be found in table 7.3. It is important that these results hold true only for the one experiment we made in this chapter. To get reliable assertions about the performance and costs further studies are necessary. However, the results of this chapter and the two-dimensional results indicate that in the three-dimensional case MLDA will outperform the MH algorithm.

Algorithm	**Burn in**	**IAT**	**Costs per sample**	**Costs per effective sample**
MH	5000	3672	$c^f + c^v_{fine}$	$3672(c^f + c^v_{fine})$
MLDA	250	455	$\leq 7(c^f + c^v_{fine})$	$\leq 3185(c^f + c^v_{fine})$

Table 7.3: Comparison of the performance of the MH and the MLDA algorithm in our experiment showing that the MLDA algorithm has less computational costs.

8 Conclusion

The aim of this thesis was to quantify uncertainties in the reconstruction of sources of measured EEG potentials by reconstructing a probability distribution of the source using the MLDA algorithm.

First, we analyzed different types of uncertainties occurring in the context of the EEG model. We found that the uncertainties in the measurement values caused by internal and external noise have the strongest impact on the results of source reconstruction.

We have shown how the MH algorithm can be applied to the task of EEG source reconstruction, considering these uncertainties by including the specific variance at each sensor. The results could be verified in the two-dimensional setting by using numerous simulated measurement values with known sources. In the three-dimensional setting we applied the algorithms to real measurement values from a somatosensory experiment. The dipole-scan method was executed to generate reference values. In the two-dimensional setting as well as in the three-dimensional setting it could be shown that the MH algorithm reconstructs the source of a EEG-measured potential reliably.

We elaborated that the MH algorithm in the context of source reconstruction has a high integrated autocorrelation time, indicating a high correlation of sequential samples. Therefore, it has to be run for a high number of steps to guarantee a correct approximation of the posterior distribution. To improve the stability of the source reconstruction and to reduce the computational costs, we introduced the MLDA algorithm as a multilevel version of the MH algorithm. We constructed different hierarchies for the MLDA algorithm by reducing the number of electrodes as well as the mesh size and switching the source model from the Venant approach to Partial Integration. In this context we have shown a method to reduce the number of sensors by executing a PCA to transform the transfer matrix. This method works remarkably well such that the number of sensors can be reduced to a quarter while keeping a high portion of the information. Furthermore, we showed how the transfer matrix can be transformed on the nodes of a coarser mesh.

In the two-dimensional experiments we compared the MH algorithm and different variations of the MLDA algorithm. The analysis of the costs of the algorithms showed that the MLDA algorithm outperforms the MH algorithm provided an efficient implementation is used. It has been found that a reduction of the number of electrodes by a half or even by three quarters in combination with the usage of Partial Integration instead of the Venant approach on the coarse level of the MLDA algorithm, leads to the best results.

Regarding the computational costs in the three-dimensional experiment, it could be shown that for the given data the MLDA algorithm has been producing less computational costs than the MH algorithm. This observation and the two-dimensional results together, indicate that in the three-dimensional case MLDA will outperform the MH algorithm in general. To get a reliable assertion on this subject further studies are necessary. In particular, a higher number of tests is required.

Moreover, there are numerous other possibilities for further studies. On the one hand the algorithms could be applied for other settings occurring in practice as for example the reconstruction of multi-focal sources. On the other hand it could be attempted to improve the posterior distribution by integrating more knowledge such as the tendency of dipoles to be locally radially oriented. In addition, further hierarchies could be constructed, complementing the hierarchies presented in this thesis.

A Parameters of the two-dimensional MLDA algorithms

In the following the chosen parameters of the MLDA algorithms executed in chapter 6.2.2 are listed to allow a repetition of the presented results.

Parameter	M1 - coarse level	M2 - coarse level
Mesh	hexahedral, 128×128	hexahedral, 64×64
Electrodes	36	36
Source model	Venant	Venant
Proposal variance	(2,2,0.1)	(2,2,0.02)
Subchain length	Uniform distributed on $[2, 20]$	Uniform distributed on $[2, 20]$
Factor of variance	4	8

Table A.1: Parameters of the MLDA algorithms M1 and M2

Parameter	E1 - coarse level	E2 - coarse level	E3 - coarse level
Mesh	hexahedral, 256×256	hexahedral, 256×256	hexahedral, 256×256
Electrodes	18	9	5
Source model	Venant	Venant	Venant
Proposal variance	(1,1,0.05)	(1,1,0.05)	(1,1,0.1)
Subchain length	Uniform distributed on $[2, 20]$	Uniform distributed on $[2, 20]$	Uniform distributed on $[2, 20]$
Factor of variance	2	2	2

Table A.2: Parameters of the MLDA algorithms E1, E2 and E3

Parameter	S1 - coarse level
Mesh	hexahedral, 256×256
Electrodes	36
Source model	Partial Integration
Proposal variance	(2,2,0.1)
Subchain length	Uniform distributed on $[2, 20]$
Factor of variance	4

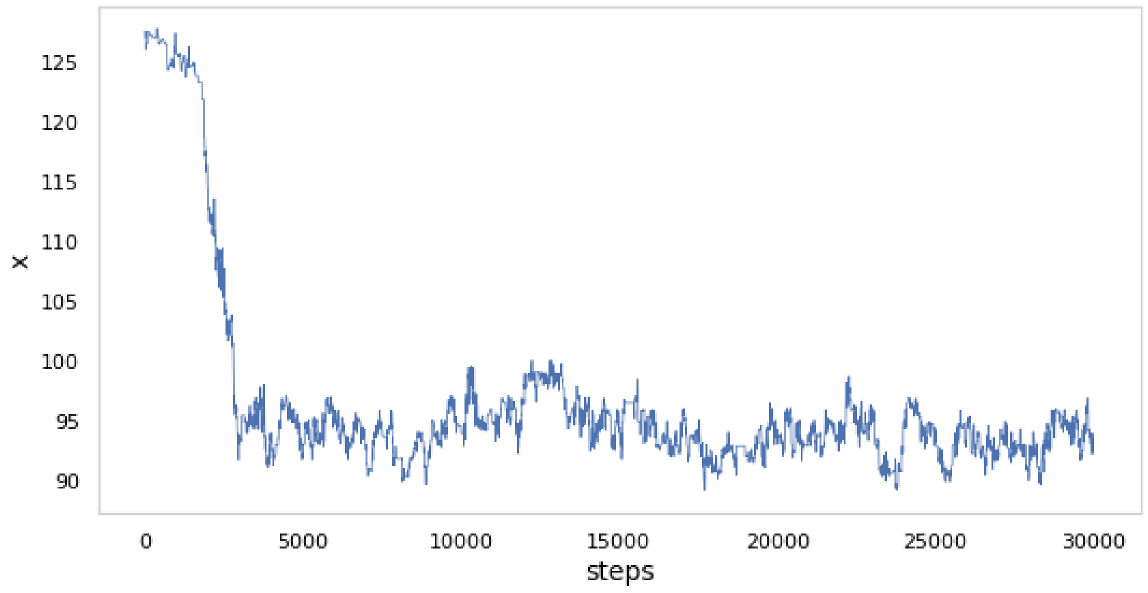
Table A.3: Parameters of the MLDA algorithm S1

Parameter	C1 - coarse level	C2 - coarse level	C3 - coarse level
Mesh	hexahedral, 128×128	hexahedral, 64×64	hexahedral, 256×256
Electrodes	9	9	5
Source model	Partial Integration	Partial Integration	Partial Integration
Proposal variance	(2,2,0.02)	(2,2,0.01)	(2,2,0.1)
Subchain length	Uniform distributed on $[2, 20]$	Uniform distributed on $[2, 25]$	Uniform distributed on $[2, 20]$
Factor of variance	4	6	4

Table A.4: Parameters of the MLDA algorithms C1, C2 and C3

B Results of the three-dimensional MLDA algorithms

In the following further results of the MH and the MLDA algorithm run in chapter 7 are shown. The trace plots show the x-coordinate of the generated samples over a number of steps.



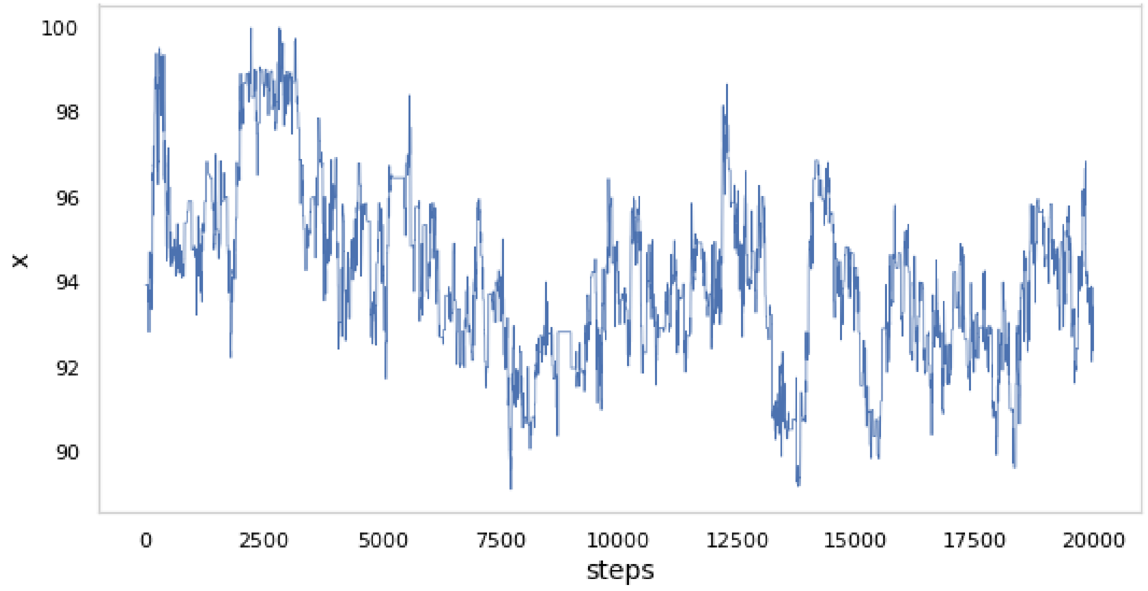
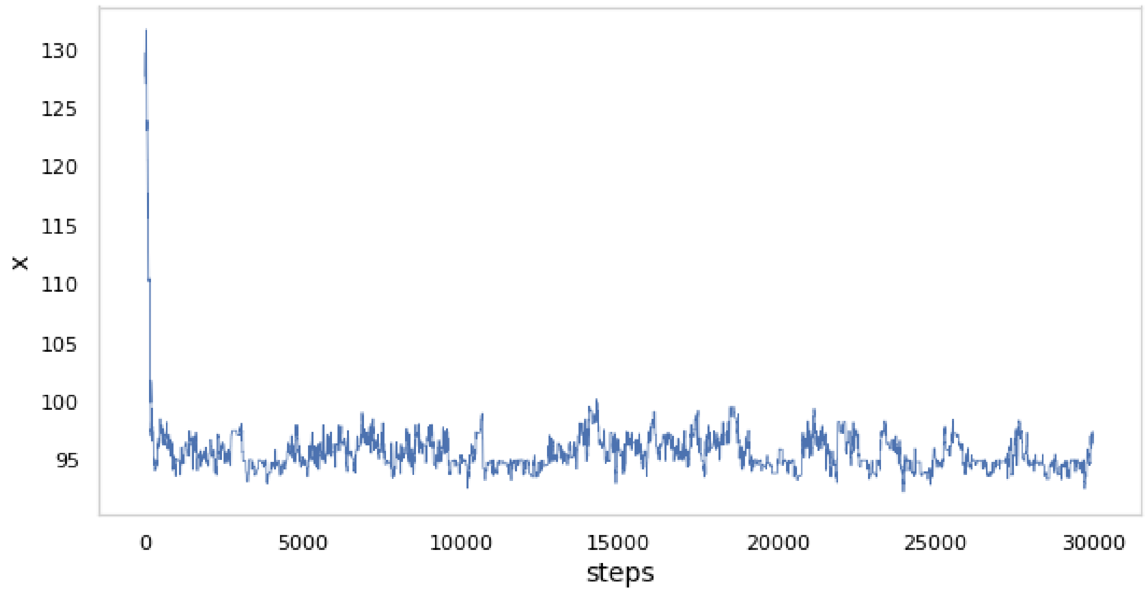


Figure B.1: Trace plots of the x-coordinate of the generated samples using the MH algorithm. The upper plot shows the first 3000 steps of the algorithm. It is visible how the chain is moving from the start point towards the dipole position. The burn in phase needs circa 3000 steps. The lower plot shows 2000 steps after the burn in phase.



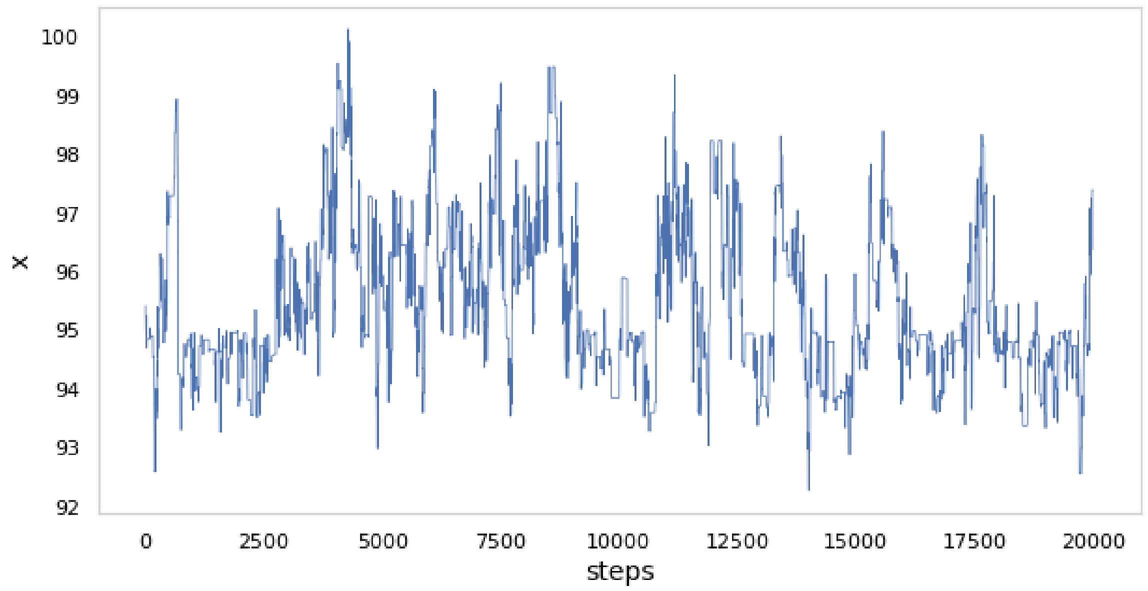


Figure B.2: Trace plots of the x-coordinate of the generated samples using the MLDA algorithm. The upper plot shows the first 3000 steps of the algorithm. It is visible how the chain is moving from the start point towards the dipole position. The burn in phase is negligible short. The lower plot shows 2000 steps after the burn in phase.

References

[1] Adnan M. Salman, Allen D. Malony, and Matthew J. Sottile. “An Open Domain-Extensible Environment for Simulation-Based Scientific Investigation (ODESSI)”. In: *Computational science*. Ed. by Allen Gabrielle. Lecture notes in computer science. Berlin: Springer, 2009, pp. 23–32. ISBN: 978-3-642-01969-2. DOI: 10.1007/978-3-642-01970-8_3. URL: https://www.researchgate.net/publication/220856618_An_Open_Domain-Extensible_Environment_for_Simulation-Based_Scientific_Investigation_ODESSI.

[2] Marios Antonakakis et al. *Inter-Subject Variability of Skull Conductivity and Thickness in Calibrated Realistic Head Models*. Vol. 223. 2020. DOI: 10.1016/j.neuroimage.2020.117353. URL: https://reader.elsevier.com/reader/sd/pii/S1053811920308399.

[3] K. A. Awada et al. “Effect of conductivity uncertainties and modeling errors on EEG source localization using a 2-D model”. In: *IEEE Transactions on Biomedical Engineering* 45.9 (1998), pp. 1135–1145. ISSN: 0018-9294. DOI: 10.1109/10.709557.

[4] S. Baillet, J. C. Mosher, and R. M. Leahy. “Electromagnetic brain mapping”. In: *IEEE Signal Processing Magazine* 18.6 (2001), pp. 14–30. ISSN: 10535888. DOI: 10.1109/79.962275.

[5] Hans Berger. “Über das Elektrenkephalogramm des Menschen”. In: *Archiv für Psychiatrie und Nervenkrankheiten* 87.1 (1929), pp. 527–570. ISSN: 0003-9373. DOI: 10.1007/BF01797193.

[6] Pierre Brémaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation and Queues*. 2nd edition. Vol. 31. Texts in applied mathematics. Cham: Springer, opyright 2020. ISBN: 9783030459819. DOI: 10.1007/978-3-030-45982-6.

[7] Marianna Brienza and Oriano Mecarelli. “Neurophysiological Basis of EEG”. In: *Clinical Electroencephalography*. Ed. by Oriano Mecarelli. Cham: Springer, 2019, pp. 9–21. ISBN: 978-3-030-04572-2. DOI: 10.1007/978-3-030-04573-9_2.

[8] H. Buchner et al. “Inverse localization of electric dipole current sources in finite element models of the human head”. In: *Electroencephalography and clinical neurophysiology* 102.4 (1997), pp. 267–278. ISSN: 0013-4694. DOI: 10.1016/s0013-4694(96)95698-9.

[9] C. Wolters et al. “Numerical Mathematics of the Subtraction Method for the Modeling of a Current Dipole in EEG Source Reconstruction Using Finite Element Head Models”. In: *undefined* (2007). URL: https://www.semanticscholar.org/paper/Numerical-Mathematics-of-the-Subtraction-Method-for-Wolters-K%C3%B6stler/9731c02d0139890f470a6eee2a807d62940662a0.

[10] Alexander J. Casson et al. “Electroencephalogram”. In: *Seamless Healthcare Monitoring: Advancements in Wearable, Attachable, and Invisible Devices*. Ed. by Toshiyo Tamura and Wenxi Chen. Cham: Springer International Publishing, 2018, pp. 45–81. ISBN: 978-3-319-69362-0. DOI: 10.1007/978-3-319-69362-0_2. URL: https://doi.org/10.1007/978-3-319-69362-0_2.

[11] Siddhartha Chib and Edward Greenberg. “Understanding the Metropolis-Hastings Algorithm”. In: *The American Statistician* 49.4 (1995), pp. 327–335. ISSN: 00031305. URL: http://www.jstor.org/stable/2684568 (visited on 06/21/2022).

[12] J. Andrés Christen and Colin Fox. “Markov chain Monte Carlo Using an Approximation”. In: *Journal of Computational and Graphical Statistics* 14.4 (2005), pp. 795–810. ISSN: 1061-8600. DOI: 10.1198/106186005X76983.

[13] B. N. Cuffin. “Effects of local variations in skull and scalp thickness on EEG’s and MEG’s”. In: *IEEE Transactions on Biomedical Engineering* 40.1 (1993), pp. 42–48. ISSN: 0018-9294. DOI: 10.1109/10.204770.

[14] Juhani Dabek et al. “Determination of head conductivity frequency response in vivo with optimized EIT-EEG”. In: *NeuroImage* 127 (2016), pp. 484–495. DOI: 10.1016/j.neuroimage.2015.11.023. URL: https://www.sciencedirect.com/science/article/pii/S1053811915010459.

[15] T. J. Dodwell et al. “Multilevel Markov Chain Monte Carlo”. In: *SIAM Review* 61.3 (2019), pp. 509–545. ISSN: 0036-1445. DOI: 10.1137/19M126966X.

[16] N. von Ellenrieder, C. H. Muravchik, and A. Nehorai. “Effects of Geometric Head Model Perturbations on the EEG Forward and Inverse Problems”. In: *IEEE Transactions on Biomedical Engineering* 53.3 (2006), pp. 421–429. ISSN: 0018-9294. DOI: 10.1109/TBME.2005.869769.

[17] Roberta Grech et al. “Review on solving the inverse problem in EEG source analysis”. In: *Journal of neuroengineering and rehabilitation* 5 (2008), p. 25. DOI: 10.1186/1743-0003-5-25. URL: https://jneuroengrehab.biomedcentral.com/track/pdf/10.1186/1743-0003-5-25%20.pdf.

[18] Gregory W. Gundersen. “Practical Algorithms for Practical Algorithms for Latent Variable Models”. In: (). URL: https://gregorygundersen.com/publications/gundersen2021thesis.pdf.

[19] M. Hämäläinen et al. “Magnetoencephalography - theory, instrumentation, and applications to noninvasive studies of the working human brain”. In: *Reviews of Modern Physics* (1993), pp. 413–497. URL: https://doi.org/10.1103/RevModPhys.65.413.

[20] Robbert L. Harms and Alard Roebroeck. “Robust and Fast Markov Chain Monte Carlo Sampling of Diffusion MRI Microstructure Models”. In: *Frontiers in Neuroinformatics* 12 (2018), p. 97. ISSN: 1662-5196. DOI: 10.3389/fninf.2018.00097. URL: https://www.frontiersin.org/articles/10.3389/fninf.2018.00097/full.

[21] Thomas Haslwanter. *Introduction to statistics with python - with applications in the life scie*. STATISTICS AND COMPUTING. Cham: Springer International Publish, 2016. ISBN: 978-3-319-28315-9. DOI: 10.1007/978-3-319-28316-6.

[22] T. Heinonen et al. “Segmentation of T1 MR scans for reconstruction of resistive head models”. In: *Computer methods and programs in biomedicine* 54.3 (1997), pp. 173–181. ISSN: 0169-2607. DOI: 10.1016/s0169-2607(97)00027-8.

[23] Li Hu and Zhiguo Zhang, eds. *EEG signal processing and feature extraction*. Singapore: Springer, 2019. ISBN: 9789811391132. URL: https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=5942859.

[24] Chang-Hwan Im. “Basics of EEG: Generation, Acquisition, and Applications of EEG”. In: *Computational EEG analysis*. Ed. by Chang-Hwan Im. Biological and Medical Physics, Biomedical Engineering. Singapore: Springer, 2018, pp. 3–11. ISBN: 978-981-13-0907-6. DOI: 10.1007/978-981-13-0908-3_1.

[25] Jan De Munck, Carsten Wolters, and Maureen Clerc. *EEG and MEG: forward modeling*. 2012. URL: https://www.researchgate.net/publication/278797469_EEG_and_MEG_forward_modeling.

[26] Xiao Jiang, Gui-Bin Bian, and Zean Tian. “Removal of Artifacts from EEG Signals: A Review”. In: *Sensors (Basel, Switzerland)* 19.5 (2019). DOI: 10.3390/s19050987.

[27] Valer Jurcak, Daisuke Tsuzuki, and Ippeita Dan. “10/20, 10/10, and 10/5 systems revisited: their validity as relative head-surface-based positioning systems”. In: *NeuroImage* 34.4 (2007), pp. 1600–1611. DOI: 10.1016/j.neuroimage.2006.09.024.

[28] Jari Kaipio and Erkki Somersalo. “Statistical inverse problems: Discretization, model reduction and inverse crimes”. In: *Journal of Computational and Applied Mathematics* 198.2 (2007), pp. 493–504. ISSN: 03770427. DOI: 10.1016/j.cam.2005.09.027.

[29] Haya Kaspi, S. P. Meyn, and R. L. Tweedie. “Markov Chains and Stochastic Stability”. In: *Journal of the American Statistical Association* 92.438 (1997), p. 792. ISSN: 01621459. DOI: 10.2307/2965732.

[30] Achim Klenke. *Wahrscheinlichkeitstheorie*. 3. Aufl. 2013. Springer-Lehrbuch Masterclass. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. ISBN: 978-3-642-36017-6. DOI: 10.1007/978-3-642-36018-3.

[31] Thomas R. Knösche and Jens Haueisen. *EEG/MEG Source Reconstruction: Textbook for Electro-and Magnetoencephalography*. 1st ed. Cham: Springer, 2022. ISBN: 978-3-030-74916-3. DOI: 10.1007/978-3-030-74918-7.

[32] Hervé Le Dret and Brigitte Lucquin. “The Variational Formulation of Elliptic PDEs”. In: *Partial Differential Equations: Modeling, Analysis and Numerical Approximation*. Birkhäuser, Cham, 2016, pp. 117–143. DOI: 10.1007/978-3-319-27067-8_4. URL: https://link.springer.com/chapter/10.1007/978-3-319-27067-8_4.

[33] Jake Lever, Martin Krzywinski, and Naomi Altman. “Principal component analysis”. In: *Nature Methods* 14.7 (2017), pp. 641–642. ISSN: 1548-7091. DOI: 10.1038/nmeth.4346. URL: https://www.nature.com/articles/nmeth.4346.

[34] S. Lew et al. “Accuracy and run-time comparison for different potential approaches and iterative solvers in finite element method based EEG source analysis”. In: *Applied numerical mathematics : transactions of IMACS* 59.8 (2009), pp. 1970–1988. ISSN: 0168-9274. DOI: 10.1016/j.apnum.2009.02.006.

[35] Jun S. Liu. *Monte Carlo Strategies in Scientific Computing.* Springer Series in Statistics Ser. Cham: Springer International Publishing, 20. ISBN: 9780387763712. URL: https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=6283375.

[36] Felix Lucka et al. "Hierarchical Bayesian inference for the EEG inverse problem using realistic FE head models: depth localization and source separation for focal primary currents". In: *NeuroImage* 61.4 (2012), pp. 1364–1382. DOI: 10.1016/j.neuroimage.2012.04.017.

[37] Mikkel B. Lykkegaard et al. *Multilevel Delayed Acceptance MCMC.* URL: https://arxiv.org/pdf/2202.03876.pdf.

[38] Takfarinas Medani et al. "FEM Method for the EEG Forward Problem and Improvement Based on Modification of the Saint Venant's Method". In: *Progress In Electromagnetics Research* 153 (2015), pp. 11–22. DOI: 10.2528/PIER15050102.

[39] Solveig Næss et al. "Corrected Four-Sphere Head Model for EEG Signals". In: *Frontiers in human neuroscience* 11 (2017), p. 490. ISSN: 1662-5161. DOI: 10.3389/fnhum.2017.00490.

[40] Frank Neugebauer et al. "Validating EEG, MEG and Combined MEG and EEG Beamforming for an Estimation of the Epileptogenic Zone in Focal Cortical Dysplasia". In: *Brain Sciences* 12.1 (2022). DOI: 10.3390/brainsci12010114. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8796031/.

[41] Andreas Nüßing. *Fitted and unfitted finite element methods for solving the EEG forward problem.* Ed. by Christian Engwer. [Electronic ed.] 2018. URL: https://nbn-resolving.de/urn:nbn:de:hbz:6-67139436770.

[42] Matthew Parno, Andrew Davis, and Linus Seelinger. "MUQ: The MIT Uncertainty Quantification Library". In: *The Journal of Open Source Software* 6.68 (2021), p. 3076. ISSN: 2475-9066. DOI: 10.21105/joss.03076. URL: https://www.researchgate.net/publication/356943825_MUQ_The_MIT_Uncertainty_Quantification_Library.

[43] Gopinath Rebala. *An Introduction to Machine Learning.* 1st edition 2019. Cham: Springer International Publishing, 2019. ISBN: 9783030157296. URL: https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=5770988.

[44] G. Repovš. "Dealing with Noise in EEG Recording and Data Analysis". In: *Informatica Medica Slovenica* 15 (2010), pp. 18–25. URL: https://ims.mf.uni-lj.si/ims_archive/15-01/15b-21.pdf.

[45] Ville Rimpiläinen et al. "Improved EEG source localization with Bayesian uncertainty modelling of unknown skull conductivity". In: *NeuroImage* 188 (2019), pp. 252–260. URL: https://www.sciencedirect.com/science/article/abs/pii/S1053811918321396.

[46] Gareth O. Roberts and Jeffrey S. Rosenthal. "General state space Markov chains and MCMC algorithms". In: *Probability Surveys* 1.none (2004). ISSN: 1549-5787. DOI: 10.1214/154957804100000024.

[47] S. Kumar and P. K. Singh. "An overview of modern cache memory and performance analysis of replacement policies". In: *2016 IEEE International Conference on Engineering and Technology (ICETECH).* 2016, pp. 210–214. DOI: 10.1109/ICETECH.2016.7569243.

[48] Robert F. Schmidt, Gerhard Thews, and Florian Lang. *Physiologie des Menschen.* Achtundzwanzigste, korrigierte und aktualisierte Auflage. Springer-Lehrbuch. Berlin, Heidelberg: Springer Berlin Heidelberg, Imprint, and Springer, 2000. ISBN: 978-3-662-09347-4. DOI: 10.1007/978-3-662-09346-7.

[49] U. Schmitt and A. K. Louis. "Efficient algorithms for the regularization of dynamic inverse problems: I. Theory". In: *Inverse Problems* 18.3 (2002), pp. 645–658. ISSN: 0266-5611. DOI: 10.1088/0266-5611/18/3/308.

[50] Sophie Schrader et al. "DUNEuro-A software toolbox for forward modeling in bioelectromagnetism". In: *PloS one* 16.6 (2021), e0252431. DOI: 10.1371/journal.pone.0252431.

[51] Storage. *What is Cache Memory? Cache Memory in Computers, Explained.* 15.01.2023. URL: https://www.techtarget.com/searchstorage/definition/cache-memory.

[52] Marco Taboga. *Autocorrelation.* Ed. by Kindle Direct Publishing. URL: https://www.statlect.com/fundamentals-of-statistics/autocorrelation (visited on 01/15/2023).

[53] Marco Taboga. *Markov Chain Monte Carlo (MCMC) diagnostics.* Ed. by Kindle Direct Publishing. URL: https://www.statlect.com/fundamentals-of-statistics/Markov-Chain-Monte-Carlo-diagnostics (visited on 01/15/2023).

[54] Magnus J. Teschner et al. “Effects of Signal-to-Noise Ratio on Auditory Cortical Frequency Processing”. In: *The Journal of neuroscience : the official journal of the Society for Neuroscience* 36.9 (2016), pp. 2743–2756. DOI: 10.1523/JNEUROSCI.2079-15.2016. URL: https://www.jneurosci.org/content/jneuro/36/9/2743.full.pdf.

[55] *UM-Bridge documentation — UM-Bridge documentation.* URL: https://um-bridge-benchmarks.readthedocs.io/en/docs/index.html (visited on 01/25/2023).

[56] Aki Vehtari et al. “Rank-Normalization, Folding, and Localization: An Improved Rˆ for Assessing Convergence of MCMC (with Discussion)”. In: *Bayesian Analysis* 16.2 (2021). ISSN: 1936-0975. DOI: 10.1214/20-BA1221. URL: https://arxiv.org/pdf/1903.08008.pdf.

[57] Johannes Vorwerk et al. “A guideline for head volume conductor modeling in EEG and MEG”. In: *NeuroImage* 100 (2014), pp. 590–607. DOI: 10.1016/j.neuroimage.2014.06.040. URL: https://www.sciencedirect.com/science/article/pii/S1053811914005175.

[58] Johannes Vorwerk et al. “Influence of Head Tissue Conductivity Uncertainties on EEG Dipole Reconstruction”. In: *Frontiers in Neuroscience* 13 (2019), p. 531. ISSN: 1662-453X. DOI: 10.3389/fnins.2019.00531. URL: https://www.frontiersin.org/articles/10.3389/fnins.2019.00531/full#h6.

[59] Johannes Vorwerk et al. “The multipole approach for EEG forward modeling using the finite element method”. In: *NeuroImage* 201 (2019), p. 116039. DOI: 10.1016/j.neuroimage.2019.116039.

[60] C. H. Wolters, L. Grasedyck, and W. Hackbusch. “Efficient computation of lead field bases and influence matrix for the FEM-based EEG and MEG inverse problem”. In: *Inverse Problems* 20.4 (2004), pp. 1099–1116. ISSN: 0266-5611. DOI: 10.1088/0266-5611/20/4/007.

[61] C. H. Wolters et al. “Numerical Mathematics of the Subtraction Method for the Modeling of a Current Dipole in EEG Source Reconstruction Using Finite Element Head Models”. In: *SIAM Journal on Scientific Computing* 30.1 (2008), pp. 24–45. ISSN: 1064-8275. DOI: 10.1137/060659053.

[62] Guoxing Zhu, Yuangui Huang, and Xuefeng Wang. “Basic Theory of EEG”. In: *Multi-Modal EEG Monitoring of Severely Neurologically Ill Patients.* Ed. by Suyue Pan, Feng Li, and Xuefeng Wang. Singapore: Springer, 2022, pp. 3–25. ISBN: 978-981-16-4492-4. DOI: 10.1007/978-981-16-4493-1_1.

[63] Johanna M. Zumer et al. “A probabilistic algorithm integrating source localization and noise suppression for MEG and EEG data”. In: *NeuroImage* 37.1 (2007), pp. 102–115. ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2007.04.054. URL: https://www.sciencedirect.com/science/article/pii/S1053811907003539.