A SOFTWARE FRAMEWORK FOR SIMULATION-BASED SCIENTIFIC INVESTIGATIONS

by

ADNAN M. SALMAN

A DISSERTATION

Presented to the Department of Computer and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

March 2010

"A Software Framework for Simulation-Based Scientific Investigations," a dissertation prepared by Adnan M. Salman in partial fulfillment of the requirements for the Doctor of Philosophy degree in the Department of Computer and Information Science. This dissertation has been approved and accepted by:

Dr. Allen D. Malony, Chair of the Examining Committee

Date

Committee in charge:	Dr. Allen D. Malony, Chair Dr. Don Tucker Dr. John Conery Dr. Dejing Dou

Accepted by:

Dean of the Graduate School

An Abstract of the Dissertation of

Adnan M. Salman for the degree of Doctor of Philosophy
in the Department of Computer and Information Science
to be taken March 2010

Title: A SOFTWARE FRAMEWORK FOR SIMULATION-BASED
SCIENTIFIC INVESTIGATIONS

Approved: ______________________________
Dr. Allen D. Malony, Chair

This thesis provides a design and development of a software architecture and programming framework that enables domain-oriented scientific investigations to be more easily developed and productively applied. The key research concept is the representation and automation of scientific studies by capturing common methods for experimentation, analysis and evaluation used in simulation science. Such methods include *parameter studies*, *optimization*, *uncertainty analysis*, and *sensitivity analysis*. While the framework provides a generic way to conduct investigation on an arbitrary simulation, its intended use is to be extended to develop a domain computational environment. The framework hides the access to distributed system resources and the multithreaded execution. A prototype of such a framework called ODESSI (Open Domain-oriented Environment for Simulation-based Scientific Investigation, pronounced odyssey) is developed and evaluated on realistic problems in human neuroscience and computational chemistry domains.

ODESSI was inspired by our domain problems encountered in the computational modeling of human head electromagnetic for conductivity analysis and source localization. In this thesis we provide tools and methods to solve state of the art problems in head modeling. In particular, we developed an efficient and robust HPC solver for the forward problem and a generic robust HPC solver for bEIT (bounded Electrical Impedance Tomography) inverse problem to estimate the head tissue conductivities. Also we formulated a method to include skull inhomogeneity and other skull variation in the head model based on information obtained from experimental studies.

ODESSI as a framework is used to demonstrate the research ideas in this neuroscience domain and the domain investigations results are discussed in this thesis. ODESSI supports both the processing of investigation activities as well as manage its evolving record of information, results, and provenance.

CURRICULUM VITAE

NAME OF AUTHOR: Adnan M. Salman

PLACE OF BIRTH: Azzoun Atmeh, Palestine

DATE OF BIRTH: November 21, 1965

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon
University of Utah
An-Najah University

DEGREES AWARDED:

Doctor of Philosophy in Computer and Information Science,
2009, University of Oregon

Master of Science in Computer and Information Science,
2001, University of Oregon

Master of Science in Physics,
1997, University of Utah

Bachelor of Science in Physics,
1992, An-Najah University, Palestine

AREAS OF SPECIAL INTEREST:

Computational Science
High Performance Computing
Neuroscience

PROFESSIONAL EXPERIENCE:

Graduate Research Fellow, Neuroinformatic Center, University of Oregon, 2004 - present

Teaching Staff, Department of Computer Science, The Arab-American University - Jenin - Palsetine, 2001 - 2003

Graduate Research Fellow, Physics Department, University of Oregon, 1999-2001

GRANTS, AWARDS AND HONORS:

Best Paper Award, International Conference on Computational Science (ICCS 2005), 2005.

Best Paper Award, International Conference on Computational Science (ICCS 2007), 2007.

PUBLICATIONS:

A. Salman, A. Malony, M. Sottile, "An Open Domain-extensible Environment for Simulation-based Scientific Investigation (ODESSI)", in *International Conference on Computational Science (ICCS 2009)*, May 2009.

S. Turovets, P. Poolman, A. Salman, K. Li, A. Malony and D. Tucker, "Bounded Electrical Impedance Tomography for Noninvasive Conductivity Estimation of Human Head Tissues", in *Electrical Impedance Tomography Conference (EIT 2009)*, June 2008, Manchester.

A. Salman, A. Malony, S. Turovets, and D. Tucker, "On the Role of Skull Parcellation in the Computational Modeling of Human Head Conductivity", in *Electrical Impedance Tomography Conference (EIT Conference 2008)*, pp. 16-18 June, 2008, Dartmouth College, New Hampshire.

S. Turovets, P. Poolman, A. Salman, A. Malony, and D. Tucker,“Conductivity Analysis for High-Resolution EEG”, in *International Conference on BioMedical Engineering and Informatics (BMEI)*, 2008.

A. Salman, A. Malony, S. Turovets, and D. Tucker, “Use of Parallel Simulated Annealing for Computational Modeling of Human Head Conductivity”, in *International Conference on Computational Science (ICCS 2007), Y. Shi et al. (Eds.),* LNCS 4487, pp. 86-93, July 2007.

A. Salman, A. Malony, S. Turovets, A. Morris, and D. Tucker, “Modeling Human Head Electromagnetics on the Cell Broadband Engine,” in *Workshop on Solving Computational Challenges in Medical Imaging, Seattle*, 2007. (Poster)

A. Salman, S. Turovets, A. Malony, P. Poolman, C. Davey, J. Eriksen, and D. Tucker, “Noninvasive conductivity extraction for high-resolution EEG source localization”, in *Advances in Clinical Neuroscience and Rehabilitation*, 6(1), pp. 27-28, March 2006.

S. Turovets, A. Salman, A. Malony, J. Eriksen, and D. Tucker. “Anatomically Constrained Conductivity Estimation of the Human Head Tissues in Vivo: Computational Procedure and Preliminary Experiments, in *7-th Electrical Impedance Tomography (EIT) Conference*, 2006.

A. Salman, S. Turovets, A. Malony, J. Eriksen, and D. Tucker, “Computational Modeling of Human Head Conductivity”, in *International Conference on Computational Science (ICCS 2005), V.S. Sundrem et al. (Eds.),* LNCS 3514, pp. 631-638, May 2005.

A. Salman, S. Turovets, A. Malony, and V. Volkov, “Multi-cluster, Mixed-mode Computational Modeling of Human Head Conductivity”, in *International Workshop on OpenMP (IWOMP)*, June 2005.

ACKNOWLEDGMENTS

This dissertation would not have been possible without the help, support and guidance of many people. First, I would like to express my deepest gratitude to my advisor Dr. Allen Malony for his excellent guidance, caring, support and for setting up an excellent environment for conducting research. His enthusiasm has been a major driving force through out the years of this research. He was always available to discuss ideas, read and response to drafts and questions even when he was physically on a different continent. His oral and written responses were always very helpful and appropriate. I appreciate his approach of letting me explore on my own while maintaining his guidance to keep me focused.

I would like to thank my committee members Dr. John Conery, Dejing Dou, and Don Tucker for their inputs throughout the process of completing this dissertation. In particular, I would like to express my greatest thanks to Don Tucker who provided me with many valuable ideas and discussions on the scientific part of this research.

I am very thankful to all my colleagues at the NeuroInformatic Center and the office manager Charlotte Wise for providing such a great research environment. In particular, many thanks to my colleague and friend Dr. Sergei Turavets who introduced me to the Human Head Modeling research and we continued to work closely together through out the years. My greatest thanks to the system staff who maintained all the clusters up and running and being prompt in responding to questions. I am also very thankful to Dr. Robert Yelle who provided me with the chemistry example which I used in the evaluation part of this dissertation. Also, I would like to thank my colleagues at the EGI Company for their significant contributions to the scientific part of this dissertation, in particular, for providing me with all the images and data which I used in this research. Especially, many thanks for Colin Davey for his quick response when needed.

My greatest thanks to the faculty, staff and students at the computer science department especially to Dr. Arthur Farley and Dr. Andrzej Proskursowski for their assistance and guidance in getting my graduate study started and introducing me to the research in computer science. My greatest thanks to Star Holmberg (the graduate coordinator) for keeping me updated with the many rules and being there when needed even after her working hours.

Most importantly, I would like to thank my parents, sisters, brothers, parents-in-law and friends for their persistence unconditional support throughout my life. To my wife Reema who walked with me step by step, I would like to express my love and say, we succeeded together. To my sons Mohammed and Razi, all of this is just for you.

The research in this dissertation was partially funded by a fellowship from the Islamic Bank at Jeddah, Saudia Arabia through the ministry of higher education in Palestine.

DEDICATION

To my mother

TABLE OF CONTENTS

Chapter Page

I. INTRODUCTION 1

1.1 Motivation 1
1.2 Contributions 6
1.3 Dissertation Organization 8

II. SCIENTIFIC INVESTIGATIONS 9

2.1 Experimental Science 10
2.2 Theoretical Science 14
2.3 Computational Science 16
2.4 Summary and Conclusions 34

III. NEUROIMAGING OF BRAIN DYNAMICS 36

3.1 Neuron Anatomy 38
3.2 Neuron Interaction 39
3.3 Cerebral Cortex 47
3.4 The Origin of EEG and MEG Signals 49
3.5 Metabolism-based Neuroimaging 51
3.6 Electromagnetic-based Neuroimaging 55
3.7 High-resolution Neuroimaging 57
3.8 Summary and Conclusions 66

IV. HUMAN HEAD ELECTROMAGNETICS 70

4.1 Maxwell's Equations 71
4.2 Quasi-static Approximation 72
4.3 Volume Conductor Models 75
4.4 Finite Difference Discretization 99
4.5 Evaluations and Results 107

Chapter Page

4.6 Summary and Conclusions 115

V. ELECTRICAL IMPEDANCE TOMOGRAPHY 118

5.1 Conductivity Estimation 120
5.2 Objective Functions 123
5.3 Global Optimization Algorithms 125
5.4 Conductivity Modeling - Simplex Search 126
5.5 Conductivity Modeling - Simulated Annealing 137
5.6 Skull Inhomogeneity 156
5.7 Summary and Conclusions 162

VI. ODESSI DESIGN 165

6.1 Introduction 165
6.2 ODESSI Requirements and Design 168
6.3 Related Work 174

VII. ODESSI DEVELOPMENT 181

7.1 Scientific Investigation Methods 187
7.2 Execution Model 212
7.3 Scientific Data Models 225
7.4 Building a Domain PSE 234
7.5 Summary and Conclusions 239

VIII. ODESSI EVALUATION 243

8.1 Computational Chemistry Domain 244
8.2 Human Neuroscience Domain 253
8.3 Summary and Conclusions 265

IX. CONCLUSION 267

Chapter Page

BIBLIOGRAPHY 270

LIST OF FIGURES

Figure Page

2.1 The three interacting areas of scientific research 11
2.2 The steps of the scientific method 12
2.3 The structure of scientific investigations 13
3.1 The structure of a neuron consists of a soma, dendrites and an axon. . . 38
3.2 (a) When the action potential reaches the axon terminal, it triggers the release of neurotransmitter substances into the synaptic gap (b) The neurotransmitter substances then bind to the receptors in the postsynaptic neuron. The binding causes a glial-activated channel to open. If the binding causes Na^+ channels to open, the Na^+ flux into the cell alters the membrane potential and produces an excitatory postsynaptic potential. If the neurotransmitter substance causes Cl^- channels to open, the Cl^- flux into the cell causes an inhibitory postsynaptic potential. 40
3.3 Neuron at rest (top). Propagation of action potential (bottom). Sodium channels open when the membrane is sufficiently depolarized. The leading edge of the action potential activates other sodium channels, and a wave of depolarization is initiated. The refractory period forces the action potential to travel only in one direction. The intercellular and extracellular currents form a complete current circuit. 41
3.4 An equivalent dipole magnetic field, volume and primary currents, and the equipotential lines. 46
3.5 The surface of the cerebral cortex. 48
3.6 Cerebral cortex: each equivalent current dipole is associated with an area of the cortical surface and points normally toward the surface. It corresponds to a synchronous sum of hundreds of millions of synaptic activities. The image on the right is adopted from [105]. 50
4.1 A current dipole moment, $\mathbf{q} = Id\mathbf{e}$, can be expressed in terms of three orthogonal dipole moments. 105
4.2 Verification of the forward solver accuracy against analytics for a 4-shell spherical model. 108
4.3 Simulating a hole in the skull. The potential distribution (left) and the current (right). 109
4.4 Current and potential distributions due to a current dipole source in the cortex; holes in the skull allow more current to flow through the skull and reach the scalp, which increases the effective conductivity of the skull. . . 109

Figure Page

4.5 Speed-up of the forward solver for different problem sizes on 8-processor (left) and a 16-processor (right) IBM machines. 112
4.6 PPE controls the time loop and SPEs do the actual computations. . . . 113
4.7 Performance of the forward solver on the Cell Be. Comparison with a shared memory implementation (left) and Speedup curve (right). 114
5.1 EIT measurements, small alternating current injected into pair of electrodes and the response is measured at the rest of the electrodes. Data acquisition set up (left) and the computational model (right). 119
5.2 In bEIT one needs to know the average regional conductivities of a few tissue types (Scalp, Skull, Brain). 120
5.3 Schematic view of the parallel computational system based on simplex search optimization. 127
5.4 Results of the inverse search. Dynamics of the individual search (left) and statistics of the retrieved conductivities for about 200 initial random guesses(right). The actual number of the solutions shown is 71, their error function is less than 1 μV. 129
5.5 Solution flow at the master controller. Inverse solution arrival to the controller are marked. 130
5.6 The electrodes map on the scalp. Electrodes marked with red circles are the current sources. Electrodes marked with green are execluded in the inverse computation as outliers. 132
5.7 The distribution of Scalp, Skull and brain conductivities as obtained from the inverse search. 133
5.8 Comparisons of the measured potentials, the calculated potentials using the average conductivities from literature and the calculated potentials using the retrieved conductivities. 134
5.9 SA-inner loop parallelism, path selection. 146
5.10 Two level parallelisms: A parallel simulated annealing algorithm on the level of the intermediate loop, each task runs the forward calculations in parallel. 148
5.11 Three level parallelisms: In addition to the parallel forward calculation, each task computes the inner loop in parallel. 150
5.12 Solution flow at the master controller. Inverse solution arrival to the controller are marked. 152

Figure Page

5.13 Anatomically parcellated-skull into 11 major bones. 154
5.14 Brain, Scalp and skull-mean conductivities as a function of the assumed number of skull parts in the inverse search, the simulated data is generated using 11-part skull. 154
5.15 Skull parcellation: a) An 11-parts anatomy-based parcellation of the skull b) A 97-parts thickness-based parcellation c) A 5-parts thickness-based parcellation formed by combining the parts in b. 157
5.16 Thickness computation: The thickness at point p located on the inner surface of the skull is the smallest thickness computed by rays casted from points located on the surface of the sphere. 158
5.17 Conductivity reconstruction: Retrieved conductivities and the parameter A, using m-parts skull (m=1–5). 163
6.1 Architecture and components of ODESSI framework. 168
7.1 The investigation script runs the execution manager in a thread and each scientific method in a thread. Scientific methods request solution from the execution manager in there execution life time. 184
7.2 Parametric study specification 193
7.3 Optimization specification interface: Optimization problem specifications and setting the objective function (left). Selecting and customizing the optimization method (right). 198
7.4 Sensitivity analysis specification interface 210
7.5 The simulation description editor: The name of the simulation, its IO parameters and the execution sites where the simulation is available for execution are specified. 236
7.6 The data object schema description editor, used to define the data object structure and metadata. 238
7.7 The data object editor, used to create and edit a data object for a given kind. The metadata editor (left), and the parameters editor (right). . . . 239
8.1 NWChem simulation specification. 250
8.2 NWChem optimization specifications. 251
8.3 The dynamics of 3 processes out of 12 processes of the parallel simulated annealing method(left). The optimal objective function values for several optimization runs using different configurations and input coeffecient files (left). 252
8.4 Head modeling sensitivity analysis specifications 258
8.5 Distribution of the electrode's sensitivity due to changes in the Brain, CSF, Skull and Scalp tissue conductivities using several configurations. . 259
8.6 Tuning the forward-solver convergence parameters. 264
8.7 The potentials at the electrodes using $1mm$ resolution geometry vs. $2mm$ resolution. 265

LIST OF TABLES

Table Page

IV.1 Skull conductivity 98
IV.2 Tissues parameters in 4-shell models[61]. 107
V.1 Anatomy-based parcellation (size in voxels and conductivity in S/m. . 155
V.2 Extracted conductivities and the parameter A. 163
VIII.1 SCF Energy optimal exponents from several runs 252

CHAPTER I

INTRODUCTION

1.1 Motivation

Computational science is now accepted as an important approach for scientific investigation, and is considered equivalent in its discovery power to theoretical and experimental science. It fuses mathematical modeling, scientific simulation, and data-driven analysis with advances in high-performance computing (HPC) hardware and software, communications, and data management to conduct computational experiments that seek to capture reality in various domains. As a young third-leg of scientific discovery, the evolution of computational science reflects not only the application of increasing computational power, but the practice and sophistication of scientific methods in a computational form. Early concerns were for access to sufficient HPC, promoting research in parallel computing, computational grids, and large-scale storage, since these were (and continue to be) seen as the vehicles for "Grand Challenge" science. In addition to HPC technology access, computational science also depends on innovations for scientific collaboration, tool/data sharing, scientific data formats, informatics, and domain-specific integration. In response, computer science research has explored problem-solving environments, web-based portals/collaboraties, workflow management systems, visualization, databses and ontologies, and many other areas, in support of computational science goals.

As the computational power, data storage size, and communications bandwidth increase, scientists are creating more powerful methods, tools, and environments for solving complex problems, based on computational science advances. This is reflected in current scientific applications which are more highly integrated, and combine data and compute-intensive processing with analysis workflows and interactive visualization. It is also reflected in the evolution of scientific problem solving from simulation of isolated, single models of phenomena to simulations that couple multiple models to predict more comprehensive and complex behavior. While many of the computer and computational science challenges are based in technology integration, there are equally important concerns of how to manage complexity in scientific investigation.

There is now a strong interest among scientists to move to the next frontier of computation and information-based discovery, what is being called *integrative science*. Integrative science is based on the belief that scientific discovery in the future will come from an increased understanding of the interrelatedness of scientific domains and from building knowledge and tools for integrative problem solving.

Here the themes of multidisciplinary and interdisciplinary science are important. *Multidisciplinary* refers to a type of integrative science where contributions from several scientific fields are necessary to approach or solve a problem. Different in intent, *interdisciplinary* refers to the relationship of more than one branch (discipline) of scientific knowledge. Clearly, these two aspects of integrative science are naturally related.

The goals of integrative science challenges the computational science research community to provide new methods, frameworks, and infrastructure for addressing multidisciplinary and interdisciplinary concerns. Computational science is, in a fundamental sense, a “bridging” discipline, initially between computer science and certain scientific fields, but more so in the future between science domains themselves. The key metric of success is *scientific productivity.* In computer science, the concept of productivity is explicably tied to higher levels of abstraction – in theory, languages, algorithms, and so on – with the goal of “raising the level” of problem thinking and problem solving. Certainly, science fields are not new to leveraging core foundations, developing new instruments and methods, and “standing on shoulders” to advance scientific knowledge. Productivity to a scientist is measured in new science discoveries. The important general question is what contributions in computational science would lead to more productive scientific investigations in the future.

Despite the remarkable achievements and continued promise of computational science, it is fair to say there continues to be an issue of how to bridge *science cultures* – domain sciences and computer science – with respect to the computational-based scientific problem solving environments that are built and applied. The thesis argues that current approaches focus too strongly on the computing technology and do not provide high-level support for the common practices and methods of which a scientist is familiar. This raises the entry point for creating new computational environments, because these lower-level tools must be developed, and limits reuse of previously developed tools across scientific domains. Furthermore, current approaches do not effectively support the capture of domain-specific knowledge – processes and data for scientific investigation – such that environments can be shared and extended within a domain, and can provide high-level interfaces for multidisciplinary use.

The research problem considered in this thesis is how to design and develop a domain-specific environment for simulation-based scientific investigation that will result in productive science. The key research concept is the capture of standard procedures to conduct and analyze (simulation-based) scientific experiments in a modular, extensible, and reusable form. We call these procedures *scientific methods* and think of the methods as generating a set of simulation experiments to run. Common scientific methods include parameter studies, comparative analysis, optimization, sensitivity analysis, and uncertainty analysis. These methods are the basis upon which activities such as verification and validation, parameter tuning, and simulation-based experimentation are built for domain application. These processes that integrate different methods are the foundation of domain scientific investigations. A *scientific investigation* then is a domain-specific discovery process that applies one or more scientific methods in its lifetime. It defines the simulation codes to use, the input data files, and post-simulation analysis and visualization.

The main research goals of the thesis in computer science are:

1. To support common scientific methods used in simulation-based science.
2. To contextualize methods for domain-specific scientific investigations.
3. To capture domain-oriented investigation processes in the environment.
4. To abstract the simulation system, thereby insulating the scientist from concerns of HPC resource usage.
5. To provide a simulation experiment history for evolving scientific investigations.

However, in isolation of a scientific domain, achievement and evaluation of these goals will be poorly informed. In this respect, the thesis research is grounded in a computational problem in human neuroscience – modeling of human head electromagnetics for dynamic brain analysis. The experience developing methods and tools in this science domain exposed many of the challenges to overcome to delivering truly productive problem-solving technology.

1.2 Contributions

Thus, the contributions of this thesis are in two fold:

- Contributions related to a domain problem in neuroscience whereby new tools and methods were used to solve a human head conductivity modeling problem.
- Contributions in developing a general environment for conducting and managing simulation-based scientific investigations.

These contributions are listed below.

Contributions to head modeling in human neuroscience

1. Development of generic methods that enable improving conductivity estimation of the human head. In particular the thesis focuses on the challenges of the bounded EIT (bEIT) inverse problem (Publications [179, 181, 180, 176]).
2. Provide a method to include skull inhomogeneity and other skull variation in the conductivity modeling of the skull. (Publication [177]).
3. Provide computational tools that enable verification and validation of head modeling. This includes an efficient HPC solution to the forward problem (Publications [179, 181, 178]).

Contributions to scientific computational environments (Publication [175])

1. An architectural design (functional and system) for an environment to support simulation-based scientific investigations based on a framework model that abstracts common scientific methods, provides standard components for investigation workflow operations, and enables investigation workflow to be programmed.
2. An approach to capture common scientific methods in a general purpose software form that can be specified for use in scientific investigations, and a realization of the approach for a particular set of scientific methods in the form of a scientific methods library.
3. A programming model for scientific investigations that provides an abstract interface to scientific methods based on method parameterization, and a realization of the model using a scripting language system.
4. A simulation optimization model that decides what simulations to conduct based on the scientific method request and the current state of the investigation results, and a realization in a simulation planning system.
5. A design of a scientific investigation management system that will maintain the evolving record of a scientific study.
6. Application and evaluation of these techniques in the neuroscience domain and computational chemistry.

1.3 Dissertation Organization

The dissertation is organized into 8 chapters. The first chapter provides a background about scientific investigations and problem solving environments. The second chapter provides a background of our domain problem in neuroscience. Chapters three and four, describe our domain research in human head modeling. Chapter six proposes a conceptual design of the framework for scientific investigations. Chapter seven provides the development and implementation of the framework as realized in ODESSI framework. Chapter eight provides an evaluation of the ODESSI framework in head modeling and computational chemistry domains.

CHAPTER II

SCIENTIFIC INVESTIGATIONS

Modern scientific discovery involves the interaction between three areas of scientific research: two traditional research areas, *experimental science* and *theoretical science*, and *computational science*, a new third area that emerged in the past two or three decades. The purpose of scientific investigation is to answer questions about how nature behaves under different conditions. This will enable development of new technologies that serve our needs in addition to satisfying our curiosity. Many questions needs answering in every domain; for example, physics concerns finding answers to questions such as, what is the origin of the universe? Neuroscience tries to answer questions like, what is consciousness? To answer these kinds of questions, scientists in every domain use all scientific methodologies in an integrative manner; i.e., findings obtained using one method drive advances and new findings using the other methods, and advances in one scientific domain drive advances in other domains. This cycle continues and science advances (Figure 2.1). Nowadays, most modern scientific investigations involve computing as well as theory and experiment. Figures 2.2 shows the elements of the basic structure of scientific investigations.

In this chapter, our main focus is on the methods used in scientific investigation in general, with a further emphasis on the computational science methodology.

2.1 Experimental Science

Experimental science is a basic, traditional scientific research method. It is always described in terms of the *scientific method* [1]. The scientific method consists of five steps: (1) make an observation, (2) ask questions, (3) form a hypothesis, (4) test the hypothesis via experimentation, and (5) evaluate data and form conclusions. However, experimental science is more complex than simply applying these steps. It involves collecting data using observational techniques. The main characteristic of experimental science is that it investigates actual reality, not a model of reality. Therefore, experimental science is always necessary to validate any model (e.g., a mathematical model) of how accurately it describes the actual science. Experimental science is primarily concerned with investigating individual effects by performing controlled experiments and identifying and isolating specific variables of some phenomena in a controlled fashion.

By controlling variables, scientists can investigate the effect of one or more variables on a phenomenon in the search for cause-and-effect relationships in nature.

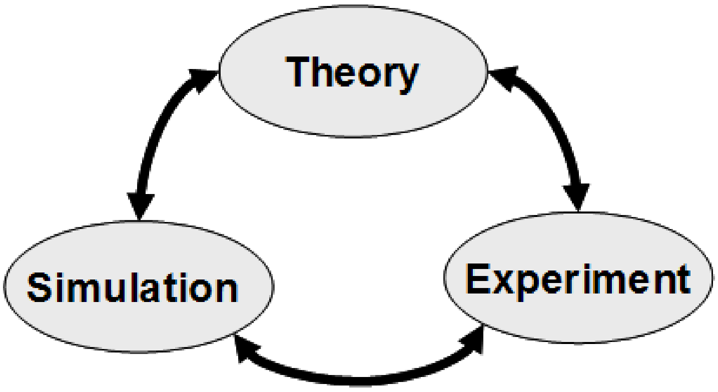


FIGURE 2.1: The three interacting areas of scientific research

The experiment is designed in a way that changes to some independent variable cause changes in some dependent variables in a predictable way. In the following discussion, we describe the basic steps of the scientific method as shown in Figure 2.3.

Asking Questions First, a scientific investigator asks a well-defined, measurable and controllable question about something that was observed. The question must be about something that can be measured.

Form a hypothesis A hypothesis is one possible explanation that answers the question. The key research is then to prove that a hypothesis is false. To do that, a researcher must design experiments. It is important that the hypothesis is stated in a way that it can be measured and evaluated.

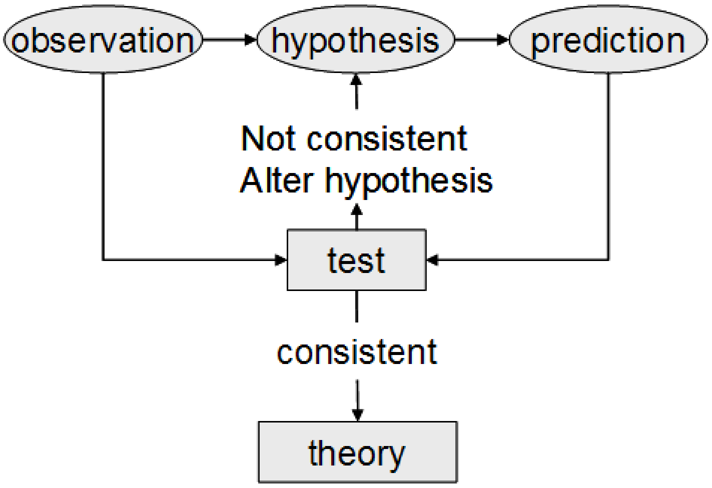


FIGURE 2.2: The steps of the scientific method

Test the hypothesis by experiment To test a hypothesis, an investigator typically sets up a controlled experiment. Constructing an experiment involves the following steps:

1. *Variables*. There are three categories of variables: *dependent*, *independent* and *controlled*. Dependent variables are what will be measured; they are what the investigator thinks will be affected during the experiment. Independent variables are variables that can be varied during the experiment–i.e., those variables that the investigator thinks may affect the dependent variables. Controlled variables are those variables that hold constant during the experiment. Since the investigator wants to study the effect of one particular independent variable, the probability that the other factors are affecting the outcome must be eliminated.

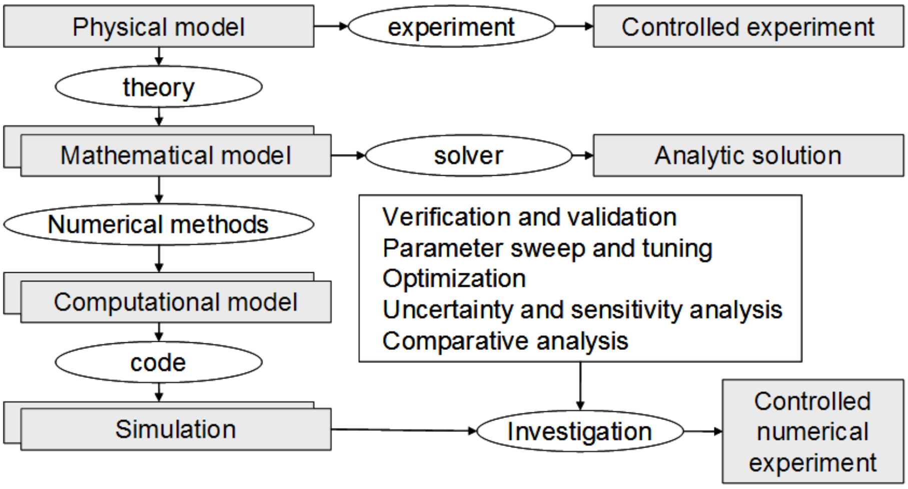


FIGURE 2.3: The structure of scientific investigations

2. *Procedures.* A procedure is the method that measures the dependent variable as a function of the independent variables. Several factors are considered in developing a procedure: for example, the appropriate values to use for the independent variable and the number of times the experiments need to be repeated to ensure that the results are consistent.

3. *Predictions.* The scientist predicts the effect of the independent variable on the dependent variable. The prediction is a statement of the expected results of the experiment based on the hypothesis. The prediction often takes the logical form of an “if/then statement.”

Data Evaluations Is the evaluation of the obtained data to determine whether or not the experiment supports the hypothesis? Sometimes statistical tests or other calculations are required to evaluate the significance of the results.

Conclusion If the experiment outcome supports the hypothesis, then the hypothesis is accepted and it becomes a theory. A hypothesis is never accepted as an absolute truth. It is accepted as long as there is no proof that it is false. If the experiment outcome contradicts the hypothesis, then either the hypothesis should be altered or a new hypothesis should be developed, and then the process is repeated.

Results communications Subsequently, the results are communicated to others in a final report. The report typically includes how the experiment was conducted, the purpose of the experiment, all required materials to carry out the experiment, a description of the experiment setup, the number of experiments, the evaluation criteria, and all the information necessary for others to repeat the experiment and verify the results.

Although experimental science is necessary to validate any model, the types of questions and investigations that can be pursued by this method are limited due to the cost of many experiments, risk of life, and physical limitations of scale such as time or space. In these cases, another approach such as theoretical or computational science is necessary.

2.2 Theoretical Science

Theoretical science is another traditional area of scientific investigation. Its main focus is in formulating a mathematical model that approximates actual reality by applying a variety of mathematical techniques. The obtained mathematical models must be validated or investigated by comparing the theoretical predictions with experimental data. The mathematical model is assumed to be correct (theory) as long as there is no experimental or observational data that violate it. Maxwell's equation, which describes the electromagnetic fields in a volume conductor, is an example of a theoretical formulation. Once a mathematical model is formulated, a complex mathematical technique is used to obtain solutions in the form of a mathematical formula. Often, several approximations are necessary to solve the mathematical model for real problems. In many cases, even with approximations, it is not possible to solve the set of equations. For example, the electromagnetic fields inside the human head due to brain dipolar current sources are formulated in terms of Maxwell's equation. To simplify the problem of solving Maxwell's equation, an investigator must apply a quasi-static approximation, which reduces the problem to solving Poisson's equation. However, considering the complex geometry of the human head, it is still not possible to obtain an exact solution to Poisson's equation, thus necessitating the approximation of the human head as multishell spheres. This leads to the third type of scientific investigation: computational science. Normally, the experimentor tries to validate the mathematics with experiments that lead to acceptance, modification, or rejection of the mathematics.

2.3 Computational Science

Computational science is the newest emerging area of scientific research. It is a methodology that allows the study of various phenomena in various domains by applying computational and numerical techniques. It has become possible due to tremendous advances in both computer hardware and software over the past few decades. Computational science (sometimes referred to as *scientific computing*, *modeling and simulation* or *simulation science*) has become a major part of every scientific domain–e.g., chemistry, physics and biology.

Nowadays, computational science is accepted as a third methodology in scientific investigations, complementing both experimental and theoretical science. It is often used to explore or validate theoretical ideas. Computer simulation is typically used to investigate phenomena that are too complex to be dealt with via analytical methods or too expensive or dangerous to be studied through experiments. Many traditional experiments that are used in scientific investigations are now replaced with simulations such as wind tunnels or nuclear fusion. Other important scientific domains involve time scales that are not possible to be investigated through experiments–e.g., astrophysics or protein folding. Many complex mathematical problems that were intractable in the past are being solved by using computational techniques.

Computational science, which involves using computers to study scientific problems, is highly related to theoretical science in that it provides solutions to complex mathematical models that cannot be solved analytically. Similar to experimental science, computational science is primarily concerned with creating and using computer models as a method of making observations, conducting controlled experiments, and defining or testing new theories. However, it is different from experimental science in that it uses an approximate model of the world instead of the world itself. Therefore, computational models must be verified and validated to build confidence in the model and evaluate how accurately it represents the science under investigation. So, computational science cannot replace experimental or theoretical science, but it is their complement in scientific investigations. It allows building models to make predictions of what might happen in the lab or in the actual world.

The Cornell Theory Center provides the following formal definition of computational science as a third methodology in scientific research: “A field that concentrates on the effective use of computer software, hardware and mathematics to solve real problems. It is a term used when it is desirable to distinguish the more pragmatic aspects of computing from (1) computer science, which often deals with the more theoretical aspects of computing; and from (2) computing engineering, which deals primarily with the design and construction of computers themselves. Computational science is often thought of as the third leg of science along with experimental and theoretical science.”

Computational science is not only a methodology of scientific research, it is a scientific domain in its own right. It is often described as an interdisciplinary approach to problem solving that uses techniques from the disciplines of (1) a domain science, (2) computer science, and (3) mathematics. Therefore, it can be defined as that science at the intersection of the three domains, so advances in computational science are therefore driven by advances in these domains. According to SIAM [191], computational science is formally defined as follows:

"Computational science and engineering (CSE) is a rapidly growing multidisciplinary area with connections to the sciences, engineering, and mathematics and computer science. CSE focuses on the development of problem-solving methodologies and robust tools for the solution of scientific and engineering problems. We believe that CSE will play an important if not dominating role for the future of the scientific discovery process and engineering design."

Application (Domain science) In computational science, the type of science under investigation is referred to as *application* or *scientific domain*. We will use both terms equivalently in this dissertation. For example, this dissertation addresses one application, computational head modeling, which involves solving a specific partial differential equation, known as Poisson's equation. Currently, computational science is used in most of the traditional scientific domains. Some popular areas of computational science include Atmospheric Science, Computational Chemistry, Computational Physics, Computational Fluid Dynamics, Computational Biology, and Nuclear Engineering. The list is much larger and continues to grow.

Algorithm (Mathematics) Computational science is similar to theoretical science in that a scientific problem must be formulated mathematically to define a mathematical model that tries to approximate the physical model. However, computational science targets problems where the solution of the mathematical model is too complex to be solved analytically. Therefore, one or more computational models in a form of algorithms are normally derived by approximating the mathematical model in a process called discretization. There are many methods used to obtain a computational model from a mathematical model. Some of the most common methods include the Finite Difference Method (FDM), Finite Element Method (FEM), and Boundary Volume Method (BVM). Most computational models use approximations and assumptions to help simplify the mathematics in a mathematical model. The computational model must be tested for how well it represents the mathematical model in a process called verification, and the mathematical model must be tested for how well it represents the actual science being modeled (physical model) in a process called validation. These evaluations normally occur throughout the modeling process.

Simulation (Computer Science) Once a computational model (algorithm) is defined, this model is implemented into a computer program (simulation). In general, the simulations are implemented by programmers working collaboratively with a domain scientist and a mathematician. Programming languages such as FORTRAN, C/C++ or Python are popular choices for most scientific simulations. In general, the simulation is implemented to run on an HPC running a variant of the UNIX operating system. How accurately the simulation implements the mathematical model is accomplished in a process called verification. Typically, the model variables (input, output, parameters) are well-defined arguments in a simulation where the user can set the input parameters and obtain the outputs after executing the simulation (conducting a controlled numerical experiment).

2.3.1 Scientific Investigation in Computational Science

Once the simulation is implemented, several scientific investigations can be performed on the simulation by conducting controlled numerical experiments. Investigations in computational science are similar to investigations in experimental science in that they involves the same five steps contained in the scientific method: ask a question, develop a hypothesis, perform an experiment to test the hypothesis, and draw conclusions. Performing a controlled numerical experiment in computational science corresponds to performing actual experiments in experimental science. Therefore, studying the cause-and-effect relationship is accomplished by varying some of the simulation input variables and measuring the response of the simulation outputs.

Also, computational science investigations differ from experimental science investigations in that computational science investigates a model of the actual science instead of the physical model itself. Therefore, before simulation can be used in scientific investigations, confidence must be established in the verification and validation processes by determining (1) how accurately the simulation implements the computational model, (2) how accurately the computational model approximates the mathematical model, and (3) how accurately the mathematical model represents the actual science.

Verification and Validation. Verification and validation (V&V) [143, 188, 189] [163][164] processes aim to characterize confidence in the computational model being used to represent the physical model. V&V has gained significant interest in recent years[190, 218], and approaches and procedures are well described [164]. Here we provide only a brief review

- **Verification** is formally defined as "the process of determining that a model implementation accurately represents the developers conceptual description of the model and the solution to the model" [143, 188]. Simply put, it deals with assessing and quantifying the confidence in the simulation, that is, in getting the equations right. It deals with the mathematics of the conceptual model rather than the physics. There are two parts to the verification process:

  - *Code verification* – provides confidence that the solution algorithm implemented in the code represents correctly the conceptual model.
  - *Solution (calculation) verification*, provides confidence that the computational model (discrete solution) is an accurate representation of the mathematical model by quantifying uncertainty.

  The verification process normally compares the computational solutions to the correct solution which is provided for well chosen analytic or manufactured solutions. The confidence in the computational solution is acceptable when the difference between the computational solution and the correct solution is small. Also, the verification assessment process involves examining the iterative, spatial and temporal convergence of the model.

- **Validation** is formally defined as “the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model” [143, 188]. Its main concern is identifying and quantifying the uncertainties between the computational model and the real world model, that is, are we solving the right equations. Validation deals with the assessment of the physical accuracy of a computational model based on comparisons between the computational simulation and experimental data[164]. Therefore, the accuracy of the validation process depends on both the accuracy of the computational model as well as on the accuracy of the experimental data.

In both computational science investigations and verification and validation processes, there are several methods and process that are common among all domains and can be reused within the domain and across domains. We call these methods *scientific investigation methods.* Some common scientific investigation methods include parameter sweep, sensitivity analysis, comparative analysis and optimization. In these processes, typically a scientist follows a process (often called a workflow) involving experiment specification, simulation runs, and results analysis and presentation. The process is then executed many times to generate outputs under different conditions (different parameters or models) based on the desired investigation. Post processing tools (e.g., visualization, statistical analysis, data mining tools) are applied to extract knowledge from the results. General methodologies can be found underlying scientific investigation across computational science domains. We summarize some of the dominant methodologies below.

Sensitivity analysis. The study of how the uncertainty in the output of a model can be a apportioned to different sources of uncertainty in the input factors is the subject of sensitivity analysis [183, 182]. Input parameters of a mathematical model are normally subject to many sources of uncertainty including errors of measurement and absence or partial information about some of these parameters. In sensitivity analysis, one tries to determine the most influential parameters that affect the global model response. Those parameters that cause significant changes in the models behavior should be sufficiently accurate prior to using the model. To accomplish this, sensitivity analysis provides means to rank input factors to their importance and so to optimize the necessary effort in getting those important factors accurate by identifying which parameters must be investigated in more details and which parameters can be removed or fixed to reduce the model number of parameters.

Uncertainty analysis. Uncertainty analysis involves probabilistic computation of the objective function due to uncertainty of some parameters. The uncertain parameters are treated as random variables and their values are taken from a particular distribution. Uncertainty analysis main focus is to quantify the uncertainty in the output. The number of simulations depends on the requirements for determining statistical significance.

Parameter sweep. Many scientific domains involve running simulation studies structured as sets of experiments in which each experiment is executed with different sets of parameters and data files. Parameter sweep analysis are used to explore the parameter space for better understanding the model behavior under different conditions before or during more complex analysis such a verification and validation. It also underlies Monte Carlo simulations. Each execution of the simulation application with distinct parameter sets is independent and can be conducted concurrently.

Optimization Optimization is an important class of scientific investigation used typically to either extract model parameters by comparing the model predictions with measured data, or to optimize some quality metric. Typically it answers the question what are the model parameters that produce an optimal results.

Comparative study Investigations are typical in computational science where there are several mathematical models (governing equations) that describe the same physical model under different set of approximations and/or there are different algorithms in solving the same mathematical model. Comparative study is the process to compare the result obtained using different models or algorithms using some metric.

These are just some of the methodologies used in simulation-based scientific investigations. They are common and thus can be applied across domains. Different domains and scientific problem investigations in those domains may use different combinations of methods. Our goal in this dissertation is to design and develop a framework to support the creation of standard methods and their incorporation in scientific investigations. In this way, our framework supports the reuse of method modules in the development of new investigation tools targeted to domain problem needs. Therefore, from a computer science perspective and in terms of scientific investigation methods, a computational science investigation can be defined as follows:

"A domain-specific discovery process that applies one or more scientific investigation methods in its lifetime. It defines the simulation codes to use, the input data files, and post-simulation analysis and visualization."

From a software engineering perspective, we can think of a computational science investigation as a script that runs and manages all the necessary computations in HPC. We call this script the domain investigation script.

2.3.2 Computational Scientist

Computational scientists are normally referred to as domain scientists or engineers who use high-performance computing to advance knowledge in their domain area of research. Typically, a computational scientist is a multidisciplinary scientist who must have enough experience in a domain scientific discipline such as physics or chemistry, and computer science, including enough experience in computer architecture, HPC, data structures, networking, massive databases, and visualization technologies in addition to expertise in numerical algorithms and statistical analysis. Therefore, a computational scientist normally conducts research at the intersection between a domain science, computer science and mathematics. Due to this integration across scientific domains, computational scientists can now approach large-scale problems that were once thought intractable [191] by integrating knowledge and methodology from all of these disciplines.

To be effective in this multidisciplinary environment, a researcher must have a background in both applications, and supporting areas of computer science and mathematics are also necessary. Our main goal in this dissertation is to provide a computational environment that will make the computational scientist more productive in conducting scientific investigation. Our domain problem in neuroscience realizes the interactions between these domains and provides a powerful example of conducting research in this multidisciplinary environment.

2.3.3 Computational Science and Computer Science

In computational science, a scientist conducts a scientific investigation to gain a better understanding of science through the use and analysis of mathematical models on computers. It is often highly associated with High Performance Computing when the computations are being performed on large computer systems. However, graphics, visualization and data storage are also important areas in modern computational science problems. So, the computer science domain is at the center of computational science. Its main focus is on the computer itself, and it involves writing software programs and the development of new hardware products. Advances in computational science, as driven by computer science, develop in two parts.

1. How to speed up computation?
2. How to make computational science researchers more productive?

The first part can be achieved by speeding up the hardware and/or by speeding up the algorithms. Following Moor's law, the power of hardware and algorithms has grown substantially in recent decades [161] [101]. On the other hand, increasing the productivity of scientist can be achieved by speeding up programming and speeding up the investigation processes. Progress in these areas has been slow in the past several decades. For instance, the productivity in writing computer code has increased by only a factor of 2-3 since 1960 [101]. Therefore, the productivity of the researcher is the main limiting factor in recent scientific advances.

Increasing the productivity of researchers in computational science can be achieved by moving the software to a higher level of abstractions. The problem-solving environment ([73, 162, 101]) is an example of movement in this direction. Problem-solving environments (PSE) are a traditional approach to addressing domain-relevant concerns by incorporating all the mathematical, algorithmic, and computational features necessary to solve a targeted class of science or engineering (S/E) problems [73, 162].

The main goal of a PSE is to increase the productivity of scientists by letting them describe a problem and its solution in terms of the S/E concepts and use a highly-functional, integrated set of capabilities for modeling, analysis, and visualization. PSEs have been developed for partial differential equations (PDE) [7], linear algebra [151], chemistry [56], and other S/E areas.

However, the traditional PSE approach has three important drawbacks: 1) it is difficult to create a new PSE, 2) PSEs are not developed to be reused, and 3) PSEs are hard to extend with new capabilities or new science methods. One response to strict PSE design is to identify domain-level functionality that is common across related fields and build software tools that can be applied in developing computational science environments [46]. Scientific development environments take this idea further by offering rich components for data management, analysis, and visualization in a programming framework for scientific applications. For example, SCIRun [193] is a powerful environment for interactive computational science which has been used to create integrated problem solving environments in biomedical science [123].

In this dissertation we propose an environment to complements these directions by abstracting common simulation-based scientific methods in reusable components, providing a cross-domain framework for scientific investigation. Web-based portals (e.g., the NEES [135] and BIRN [24] portals) and environments such as ViroLab [210] address some of these issues by offering higher-level S/E services (e.g., analysis, data management, simulation) while hiding backend complexity. The ability to abstract and reapply scientific methods for new scientific investigations or new scientific domains in these environments though is not supported well.

On the other hand, there is wealth of toolkits for scientific methods used in simulation. The DAKOTA toolkit [157] is a rich C++ toolkit that provides several optimization algorithms, uncertainty quantification, and parameter estimation.The Portable, Extensible Toolkit for Scientific Computation (PETSc) [152] is a suite of data structures and routines for the scalable (parallel) PDE-based scientific applications. The important aspect of these systems is their embodiment of a known scientific methodology in a programmable form. The idea behind ODESSI's approach is to provide a high-level scientific development framework that parameterizes and configures scientific methods for domain specialization.

In the grid environments, workflow management systems[76] have an important role in developing applications that utilize the grid available resources to conduct scientific experiments. Several Grid-enabled workflow systems are evolved in the past 10 years. Pegasus [52] is a framework that maps abstract workflow into concrete workflow and it schedules the concrete workflow into distributed resources. Triana [36] is a workflow data analysis distributed environment based on P2P, grid services and web services interaction. Tavera [145] is a service-oriented workflow system in bioinformatics where the components of the application are web services. Condor/DAGMan [67] is a resource management system used to match grid resources to tasks. Kepler [122] is an actor-oriented workflow system. WebFlow [6] and GridFlow [110] and several other PSEs are developed to ease the development of large scale scientific application from a pool of components assembled as a DAG based workflow.

These environments have an important role in building a scientific problem-solving environment that utilizes the necessary computational resources. However, these environments support only conducting a single scientific experiment or allow a parameter sweep through scheduling. What is missing in these environments is an abstract layer to allow the design of scientific studies composed of several experiments. The user is still required to conduct and manage these studies manually.

Other environments are focused on providing interactivity via parallel simulation through visualization and steering. SCIRun/UINTAH [193, 226] is a popular bioinformatics problem-solving environment that allows rapid interactive design of a scientific experiment. It also provides interactive visualization and steering. CUMULVS [75] is middleware that allows a programmer to connect remotely to a running simulation, receive visualization data, and steer user-defined parameters. gViz [99] is a grid-enabled visualization tool. In these environments, a scientist is still required to manually construct and manage the scientific investigation as a composition of several scientific experiments.

In the grid environment, there is little work that supports computational science investigations. Most of this work is limited only to parameter study or parameter sweep by generating several instances of the program corresponding to different parameters and executing these instances concurrently on the Grid or the distributed environment. Nimrod and Cluster [47] are environments that are able to generate and launch parameter study in the grid environment. They are built on top of Globus. They also provide a high-level language for describing parameter study creation. ILab [222] from NASA is a graphical tool that supports large parameter study. ILab generates a single shell script for each run in the parameter study. A single directory is created for the whole parameter study, and then a subdirectory is created for each run where input files are moved to that directory and then the scripts are executed. In the case of cluster computing, two scripts are generated, the first script remote-copies the second script to a remote cluster and then it executes there. Similar to Nimrod, AppLeS (Parameter sweep template) [29] is a parameter sweep environment. Its main focus is on scheduling the application on the grid resources in performing the parameter sweep. Similarly, Saleve [132] provides a parameter sweep across distributed resources. P-GRADE [112] portal integrates a parameter study with a workflow. It provides a parameter study by considering the workflow as a black box that gets executed with many different parameter sets. In P-GRADE the portal generates the workflow instances and submits them for execution concurrently. Another environment that integrates parameter study with workflow is SciDC [31]. MCell is a grid-enabled parameter sweep application for a biology application [28].

In all of these environments the parameter sets are pregenerated and then the response corresponding to these sets is computed. In an ongoing effort in [212], they extend this by proposing an interactive parameter sweep where the user is able to monitor and guide the parameter sets based on intermediate results. However, this approach requires the availability of the user, which is impractical in long-running simulations.

There is little work that supports other kinds of scientific investigations. Nimrod/O [2, 3] is an optimization framework that supports identifying the engineering design point by using several optimization algorithms. Sim-X [223] is an ongoing effort to provide an interactive optimization tool that allows changing the optimization criteria dynamically and exploring parts of the parameter domain while the simulation is executing. Later, SimX was added to SCIRun PSE [224]. Again, interactive optimization is not suitable for problems that require long execution times.

Most of the research work above is limited to parameter study and in few cases to optimization or stand-alone applications. These environments are closely tied to some grid infrastructure. Our goal is to extend and generalize this work and provide framework support to develop scientific investigations in a way that can draw on standard methods. As discussed in Chapters VI VII VIII the ODESSI framework will enable method implementation as programmable modules and their coupling with a simulation planning capability. Parameter sweep, uncertainty quantification, V&V, and comparative methods will be developed. Moreover, ODESSI will provide additional support for investigation scripting and access to database, analysis, and visualization utilities.

2.4 Summary and Conclusions

This chapter reviews the major methodologies used in scientific investigations and discusses the need to increase the productivity of computational scientists in solving simulation-based scientific problems in computational science. Current attempts to increase the productivity in conducting scientific investigations have proceeded mainly through building domain-specific environments. This dissertation proposes a framework that makes building such a domain environment easier by factoring out the common components of these environments (e.g., common scientific methods, simulation execution, data management) in a generic, domain-independent framework that can be customized and extended with domain-specific needs. Chapter VI discusses the design of such a framework. Chapter VII discusses ODESSI as a realization of the framework. Chapter VIII provides evaluation of the ODESSI framework on the head-modeling and chemistry domains.

The inspiration behind the design of this framework is to increase our productivity in conducting scientific research in the head-modeling domain.

Chapter III introduces our head-modeling research and the problems we are solving. Chapters IV and Chapter V discuss our contributions and our research in this area.

CHAPTER III

NEUROIMAGING OF BRAIN DYNAMICS

The human brain is the central part of the nervous system and the most complex structure in the human body. It processes information about the whole body and organizes our daily life. Brain research has been conducted for a long time for different purposes and in multiple domains, such as medicine and psychology. The foundation of neuroscience is identifying the mechanisms underyling how the brain functions, receives stimulation, and processes and stores information. An important problem in neuroscience is observing and monitoring the dynamics of the functional activities of the human brain. This knowledge is valuable in several applications across multiple domains, including the study of neural processes and the treatment of neurological disorders such as epilepsy, depression, and Parkinson's disease.

When the brain is stimulated and information is being processed, neurons in the brain active regions are activated. Activated neurons produce small currents that generate more electric and magnetic fields, consume more energy, and cause more local hemodynamic changes than inactive neurons do. Therefore, it is possible to determine the brain active regions by measuring and interpreting these effects. The science of measuring and interpreting the electric/magnetic fields from the scalp is called electroencephalography (EEG)/Magnetoencephalograpy (MEG), respectively. The EEG electric potential is measured with electrodes attached to the scalp, and the MEG magnetic field is measured with magnetic detectors placed outside the scalp. On the other hand, techniques such as Positron Emission Tomography (PET) [35] and functional Magnetic Resonance Imaging (fMRI) [209, 155] are based on measuring and interpreting the metabolic and hemodynamic changes associated with neural activity.

Each of these techniques has its own strengths and weaknesses. PET and fMRI typically produce brain functional images with high spatial resolution (1-3 mm in 3D) and poor temporal resolution (in the order of minutes in PET and about one second in fMRI). In contrast, electromagnetic-based techniques provide a high temporal resolution (order of milliseconds) but poor spatial resolution (order of a centimeter) and are less sensitive to deep sources (2D). Recent research is focused on combining some of these techniques to improve the temporal and spatial resolution of localizing brain activities [81, 195, 65]. Many factors are considered in evaluating these methods, including noninvasiveness, low cost, efficiency, and more important, reliability and accuracy.

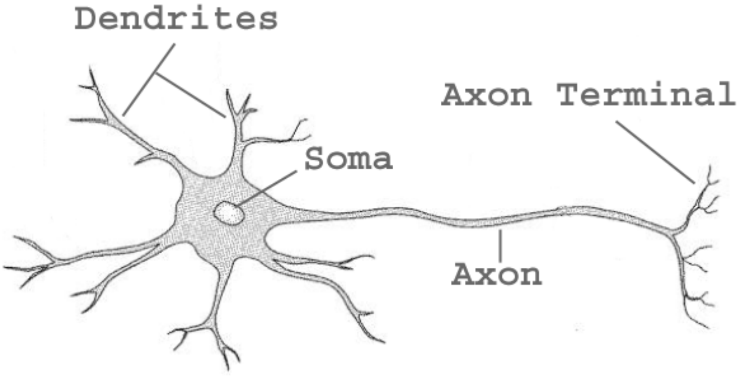


FIGURE 3.1: The structure of a neuron consists of a soma, dendrites and an axon.

In this chapter, we review the mechanism of neurons' interactions and the basics of these methods, with a focus on EEG measurements, their limitations, comparisons between them, and how to achieve high-resolution functional brain imaging.

3.1 Neuron Anatomy

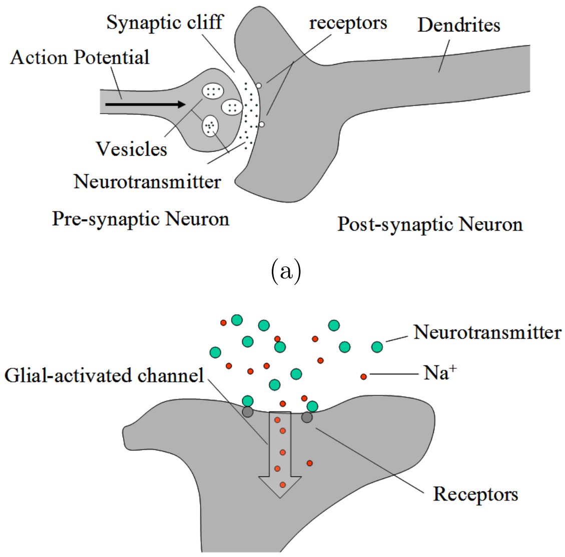
The brain is comprised of many different cells including *neurons* and *glial cells*. Neurons are the main cells that provide the fundamental functions of the brain. They are longer and thinner than other body cells. The brain contains more than 100 billon neurons, which form about 10% of the brain cells, two-thirds of them in the cerebral cortex. Most of the brain cells (90%) are glial cells, which don't carry signals but provide support functions for the neurons.

Neurons exist in different shapes and sizes. However, the main structure of a neuron consists of a cell body (soma) with branching dendrites (receivers) and an axon (transmitter) as shown in Figure 3.1. The soma contains the cell nucleus and regulates the neuron functions. The neuron senses and receives information through dendrites. The axon is a fiber that conducts information (electric impulses) to other neurons or tissues. It is coated by an insulating myelin sheath, which increases the speed of the signal and prevents signal decay. At the end of the axon is the axon terminal tree.

3.2 Neuron Interaction

The complexity of brain information processing manifests in the way neurons interact with each other. Neurons are electrically active cells. They interact with each other through connection points called *synapses* (Figure 3.2). Each synapse has two terminals. A presynaptic terminal corresponds to the axon terminal of the presynaptic neuron, and a postsynaptic terminal corresponds to the dendrite of the postsynaptic neuron. The region between the two terminals is called the *synaptic cleft*.

Neurons communicate with each other and with other cells by electrochemical signals (Figure 3.2). The chemical signals are in the form of neurotransmitter substances, while the electrical signals are in the form of impulses or action potentials that propagate along the neuron axon or dendrite. The sending



(a)

(b)

FIGURE 3.2: (a) When the action potential reaches the axon terminal, it triggers the release of neurotransmitter substances into the synaptic gap (b) The neurotransmitter substances then bind to the receptors in the postsynaptic neuron. The binding causes a glial-activated channel to open. If the binding causes Na^+ channels to open, the Na^+ flux into the cell alters the membrane potential and produces an excitatory postsynaptic potential. If the neurotransmitter substance causes Cl^- channels to open, the Cl^- flux into the cell causes an inhibitory postsynaptic potential.

neuron releases neurotransmitter substances into the synaptic cleft and the receiving neuron binds with them as shown in Figure 3.2(a). Binding with neurotransmitter substances causes the generation of postsynaptic potential (Figure 3.2(b)). If the accumulation of all postsynaptic potentials due to many synaptic activities exceeds a threshold, the neuron is activated and fires an action potential along its axon. Arrival of the action potential at the axon terminal triggers the release of neurotransmitter substances into the synaptic cleft.

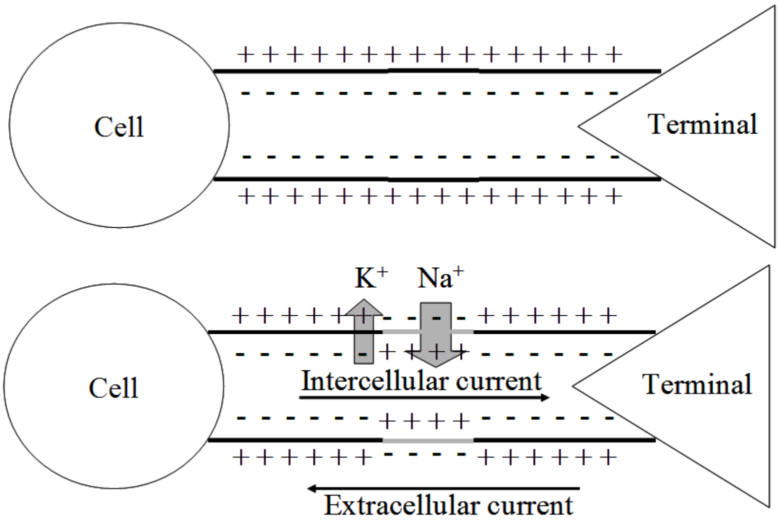


FIGURE 3.3: Neuron at rest (top). Propagation of action potential (bottom). Sodium channels open when the membrane is sufficiently depolarized. The leading edge of the action potential activates other sodium channels, and a wave of depolarization is initiated. The refractory period forces the action potential to travel only in one direction. The intercellular and extracellular currents form a complete current circuit.

Neurons receive signals asynchronously from one or many neurons and send signals to one or many neurons through the synapses' connection points. Given that there are 100 billion neurons and each neuron has thousands of synapses, this makes the brain an extremely complex network. The following subsections discuss the mechanism of these interactions in more detail.

3.2.1 Electrochemical Reactions

Like other body cells, a neuron cell has a membrane that divides the space into two regions: the intracellular region and extracellular region. The complexity of the neuron interaction with its surroundings and its behavior are determined by the complex nature of the neuron membrane. The cell membrane is not uniform; for example, some regions release neurotransmitter substances, while others bind with them. Within the membrane, there are different kinds of ion-gated channels. These channels are ion-selective. They open and close, responding to signals from the surrounding environment. Voltage-sensitive channels open and close in response to changes in the membrane potential, while ligand-gated channels open and close in response to binding with some substances (such as neurotransmitter). When a channel is open, the corresponding types of ions are allowed to cross the membrane down their concentration gradient. The rate at which ions diffuse is determined by the ion concentration difference and the electrical potential difference across the membrane. These ion-gated channels allow a neuron to receive and transmit electrochemical signals from/to another neuron or tissue in the form of action potential and electrical impulse.

The main ions involved in the electrochemical interaction of a neuron are sodium (Na^+), potassium (K^+) and chloride (Cl^-), Figure 3.3. In the rest state of a neuron (i.e., when no signal is transmitted), the extracellular concentration of sodium and chloride ions is higher than their intracellular concentration while intracellular concentration of potassium ions is higher than its extracellular concentration. Ion concentration differences are maintained by the permeability of the membrane and special ion-pumps. In the rest state, the permeability of the membrane to Na^+ is much lower than its permeability to K^+ and a special

$Na^+ - K^+$ pump pumps three Na^+ ions from the inside to the outside and two K^+ ions from the outside to the inside in each cycle. When all forces are balanced, the net potential difference across the membrane is -70 mV, where the interior of the cell is more negative. This potential difference is called the *rest state potential*.

3.2.2 Postsynaptic Potential

Postsynaptic potentials are local changes in the membrane potential of the postsynaptic terminal from the rest state potential. Dendrites are postsynaptic terminals that have receptors and are rich with ligand-activated channels. These channels open when the receptors bind with neurotransmitter substances and close gradually with time.

Depending on the type of opened channel, the corresponding ions flux across the membrane along their concentration gradient, causing local changes in the membrane potential *postsynaptic potential* (PSP). PSPs are graded potentials whose strength depends on the strength of the stimulus (the number of stimulated receptors). If the result is a current flow into the cell (in the case of an opened sodium channel), the membrane is depolarized (becomes more positive) and the PSP is *excitatory postsynaptic potential* (EPSP). If the result is a current flow out of the cell (in the case of chloride or potassium channels), the membrane is hyperpolarized (becomes more negative) and the PSP is *inhibitory postsynaptic potential* (IPSP). EPSP increases the chances that the neuron will fire an action potential, while IPSP decreases these chances. A neuron fires an action potential when the membrane sufficiently depolarizes (i.e., more than 15mV). One single EPSP cannot sufficiently depolarize the membrane to start an action potential; however, the spatial and temporal accumulation of multiple EPSP make such a depolarization possible.

3.2.3 Action Potential

A neuron transmits information to other neurons or tissues along its axon in the form of action potentials. Action potential is a wave of electrochemical activities that allow a signal to travel along the neuron axon (Figure 3.3).

The cycle of action potential is determined by the voltage-activated ion channels. When a neuron is stimulated, excitatory and inhibitory postsynaptic potentials arrive synchronously at the hillock (the region near the cell) from the synaptic activities. If the sum of all PSPs (EPSPs and IPSPs) hyperpolarizes the membrane, the action potential does not start and eventually the membrane potential returns to the rest state potential. If the sum depolarizes the membrane, Na^+ and K^+ channels open. Consequently, Na^+ ions diffuse into the cell and K^+ ions diffuse out of the cell along their concentration gradient. If the depolarization isn't strong enough (less than a threshold of 15 mV), K^+ outflow current overwhelms the Na^+ inflow current and the membrane repolarizes again. But if the depolarization is strong enough (above the threshold of 15 mV), the Na^+ current is stronger, which causes further depolarization, which in return causes more Na^+

channels to open. This sequence of depolarization drives the membrane potential to rise up. This is the rising phase of the action potentials.

When all Na^+ channels are fully opened, the membrane potential reaches its peak close to the Na^+ equilibrium potential of 55 mV. This is the end of the rising phase and the beginning of the falling phase in which Na^+ channels start to close. When some Na^+ channels close, the membrane potential falls down, and this causes more channels to close. This process continues until all Na^+ channels are closed. Since K^+ channels are still open, the membrane potential overshoots and becomes more negative than the rest state potential. When the K^+ channels close the potential eventually stabilizes to the rest state potential.

After each cycle of the action potential, Na^+ and K^+ channels require a sufficient period to recover before they are able to open and close again. This period is called *refractory period. Absolute refractory period* is the period required for most of the channels to recover. In this period the neuron can't fire new action potentials. The *relative refractory period* is the period during which enough channels are recovered to initiate a new action potential. However, in this case firing new action potential requires a higher than normal depolarization stimulus (for example, 30 mV instead of 15 mV).

In summary, an action potential starts at the hillock then propagates as a wave along the axon. The inward current at some points on the membrane provokes the nearby regions to depolarize. This depolarization provokes its neighbor channels

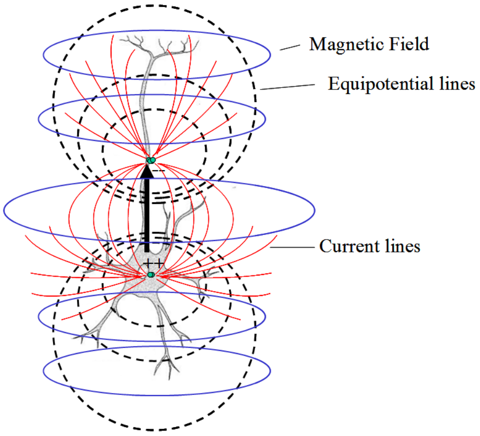


FIGURE 3.4: An equivalent dipole magnetic field, volume and primary currents, and the equipotential lines.

to open in a similar way and so on. Since the very recently opened channels are in their absolute refractory period, this guarantees that the signal will travel in only one direction along the axon where neighboring channels are not in their refractory period.

No matter how strong the stimulus is, the amplitude of the action potential remains the same. So a neuron either fires the full action potential when the membrane depolarizes beyond the threshold or it doesn't fire at all. However, the number of action potentials fired per second can differ, a variable that reflects the strength of the signal. Therefore, action potential is comprised of frequency-modulated signals.

3.2.4 Current Dipoles

While postsynaptic current propagates along the dendrites to the cell body, the postsynaptic potential decays rapidly. The result, in the case of EPSP (the opposite in the case of IPSP), is that the net external positive charge near the synapse is largest and the net external positive charge near the cell body is smallest. This creates an external voltage difference that, from a distance, looks like a current dipole oriented along the dendrite [211, 89]. To prevent accumulation of charge, an ohmic current called *volume current* in the surrounding tissues rises up and completes the current loop as shown in Figure (3.4) [89].

A current dipole is a simple convenient mathematical abstraction that represents a short spike of current. It is normally accepted to represent a biological current source when a small region of active tissues is far from the measurement sensors. Therefore, a current dipole representation is often satisfactory to explain the relationship between neuronal activity and measured fields. Higher order dipoles fall off quickly and their contribution can be ignored [211].

3.3 Cerebral Cortex

The cerebral cortex is the uppermost layer of the brain, referred to as "gray matter" due to its color. Two to four *mm* thick, it is formed by neurons. The surface of the cerebral cortex is folded in a complicated, convoluted fashion

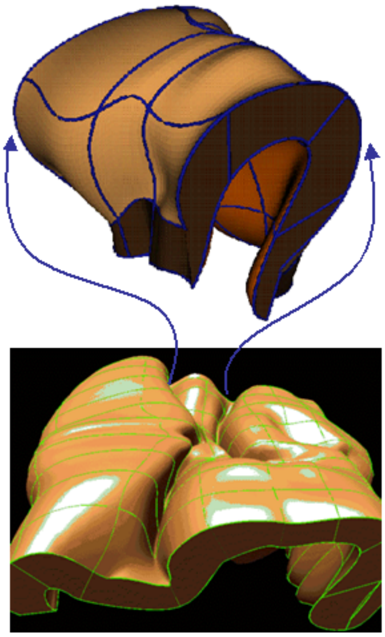


FIGURE 3.5: The surface of the cerebral cortex.

(Figure 3.5). Two-thirds of the cortex surface is buried in grooves called *sulci.* The bumps on the surface are called *gyrus.* The cerebral cortex is highly developed and responsible for thinking, language, understanding, information processing, and most human activities. The cerebral cortex is divided into lobes. Each lobe has mapped brain functionality. The four major lobes are Frontal, Parietal, Temporal and Occipital. For example, the motor cortex is responsible for the movement of a specific part of the body. Most measured neural activities take place in the cerebral cortex; therefore, this is the most relevant part of the brain, as indicated by EEG data. The density of neurons in the cerebral cortex is estimated to be 100,000 neuron/mm^2. Therefore, activation of $1mm^2$ of the cerebral cortex means the activation of about 100,000 neurons. Since each neuron has thousands of synapses, this results in activating hundreds of million of synapses.

Neurons are pyramidal cells. Their cell bodies and dendrites are located in the gray matter, while their axon extends to the white matter, where they connect different cortical areas together and provide connections between the cortex and other body parts. They are organized in groups, parallel to each other and pointing normally toward the surface. When thousands of neighboring neurons are simultaneously active, the sum of their postsynaptic potentials generates a localized current parallel to the group. This primary weak current, called an impressed current, can be strong enough to generate detectable fields on the scalp. The inverse problem of EEG/MEG is to estimate this from external observations of the fields outside the head.

3.4 The Origin of EEG and MEG Signals

It is believed and well accepted that the currents behind EEG and MEG are those corresponding to current dipoles associated with postsynaptic activities. The calculations in [89] reveal that a single postsynaptic potential produces a current-dipole moment in the order of $20fAm$ ($femto = 10^{-15}$). At least a current-dipole moment in the order of 10 nAm is required to generate detectable extracellular fields. Therefore, EEG and MEG signals are the results of the sum of at least a million simultaneously activated synapses. This sum is possible for two

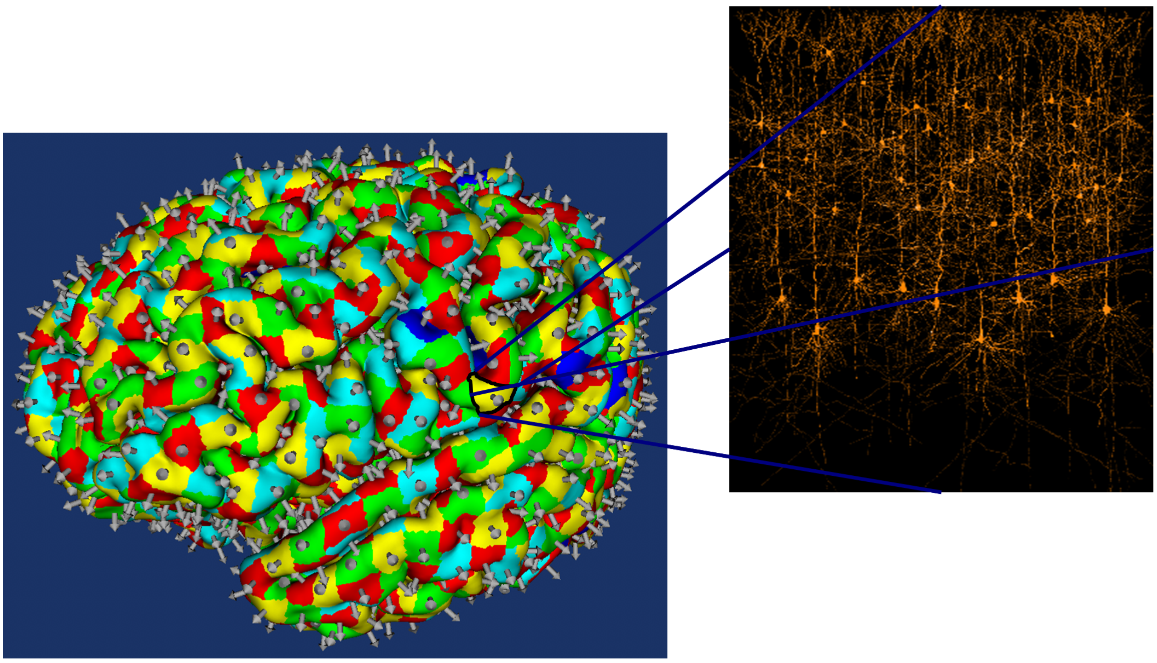


FIGURE 3.6: Cerebral cortex: each equivalent current dipole is associated with an area of the cortical surface and points normally toward the surface. It corresponds to a synchronous sum of hundreds of millions of synaptic activities. The image on the right is adopted from [105].

reasons. First, the large time course of the PSP (order of 10-20 ms) allows currents generated by synchronized synaptic activities from neighboring neurons to accumulate. Second, cortical neurons, the main generators of EEG and MEG, are arranged parallel to each other and point perpendicular to the cortical surface [141] (see Figure 3.6). This structural arrangement allows currents from groups of thousands of neurons to accumulate rather than cancel out. For instance, one mm^2 of the cortical surface contains about 100,000 neurons with thousands of synapses per neuron. Therefore, activation of a small area of the cortex can produce measurable EEG and MEG signals. Based on data from measurements, it is estimated that activation of 40 mm^2 of the cortex can produce an equivalent dipole of 10 nAm [89, 32], which is sufficient to produce measurable EEG and MEG signals.

Even though an action potential amplitude (70-110 mV) is about ten times larger than a PSP potential (1-10 mV), it is believed that action potentials have small or no contribution to the scalp fields. This is due to their short time course of about less than .3 ms, which makes asynchronous accumulation of currents from neighboring neurons unlikely. Therefore, an equivalent current dipole corresponding to a region of millions of synchronized and simultaneous PSP activities is a satisfactory, acceptable model for representing the current source behind EEG and MEG signals.

3.5 Metabolism-based Neuroimaging

These techniques are based on measuring and interpreting metabolic and hemodynamic changes associated with the brain active regions. PET is a type of nuclear medicine imaging. It is used to visualize the locations of biochemical changes such as metabolism in the body. The PET technique uses a tiny amount of radioactive organic substances (tracers or probes) to evaluate the metabolism of the brain. When a radioactive isotope (such as oxygen or carbon) is injected into the blood stream, it is consumed by the metabolically active regions. When the radioactive material decays it produces a neutron and a positron. The positron and electron annihilate and produce two gamma photons in opposite directions. These photons are then detected by a PET scanner and the position of their source is determined. These positions are then used to construct images of dynamic changes in the spatial distributions of the tracers.

The main advantage of this method is its ability to determine the active brain regions with high spatial accuracy. These images have a spatial resolution as high as $2mm$. However, this method has several main drawbacks: (1) the limitation of temporal resolution, usually a few minutes, by the dynamics of the processes being studied and photon-counting noise; (2) the use of radioactive material; and (3) its expensive apparatus and operation.

A more recent noninvasive technique is fMRI. fMRI is based on the assumption that functional activation of the brain can be detected with MRI via direct measurements, blood-volume changes, or changes in the concentration of oxygen. The most common fMRI technique that has been used in neuroscience research is the Blood Oxygen Level Dependent (BOLD) method [144, 121, 16]. The BOLD method is based on the assumption that neurons don't store reserves of energy and oxygen, so active neurons require more oxygen from the bloodstream more quickly, as compared to inactive neurons. Therefore, the blood supplies active neurons with oxygen at a higher rate compared to inactive neurons. The difference in magnetic susceptibility between the oxygenated and deoxygenated blood can be detected using an MRI scanner in which the fMRI signal is triggered by the metabolic demands of increased neural activity. The oxygenation levels can then be imaged as a correlation to the neural activity.

Compared to PET, fMRI operates at finer spatial and temporal resolutions. It can produce images with spatial resolutions as high as one mm, and a temporal resolution of approximately one second. Further, it is fully noninvasive and is relatively less expensive. Although fMRI provides improved temporal resolution compared to PET, it is still limited by the relatively slow hemodynamic response (approximately one second) when compared to electrical neural activation (which is measured in milliseconds). In addition to this limitation, interpretation of fMRI data is limited by the complex relationship between the BOLD changes detected by the fMRI signal and the underlying neural activities. Brain regions of BOLD changes do not necessarily correspond to neural activities. It might be influenced by nonneural activities.

Due to these improvements over PET, fMRI gained significant attention and generated high expectations in the neuroscience community [16], becoming the most widely used method in cognitive neuroscience. By the year 2008, over 19,000 articles were published about fMRI. For instance, Gazzaniga and Heatherton [74] described it as a “biological revolution” in psychology. However, there are a growing number of criticisms that address the method’s limitations [166, 207]. Most concerns are about the indirect relation between the fMRI signal and neural activity [94]. For example, Heeger and Ress [94] discussed this relation and pointed out the weakness of the “linear model,” which interprets the strength of the fMRI signal as proportional to local neural activity that has been averaged over a space and time. They evaluated the linear model and concluded that it is a reasonable approximation, but only for some recording sites in some brain areas under certain experimental protocols. Further, they discuss the dependency of fMRI results on the acquisition technique (BOLD results differ from non-BOLD methods).

Another article [54] discusses several neuroscientists’ concerns about the use of fMRI, and the reliability and validity of the conclusions based on the fMRI data. The articles quoted several researchers referring to fMRI image data as “gross,” claiming that the localization of cognitive functioning is not consistent with the notion that brain activity is distributed in neural networks [54]. Some researchers even questioned the quality of the images produced by fMRI.

A recent Nature article [121] provides a review of the method and its limitations, arguing that fMRI is not and will never be a mind reader, as proponents suggest, nor is it as worthless or worthy of condemnation as some have argued. The author indicates that the extreme positions on both sides result from a poor understanding of the actual capacities and limitations of the method. He points out that the limitations of the method are primarily due to the circuitry and functional organization of the brain and much less to limitations imposed by the existing hardware or the acquisition methods. Also, he indicates that the fMRI signal cannot differentiate between function-specific processing and neuromodulation and may confuse excitation and inhabitation. Therefore, the interpretation of the fMRI result is still controversial.

3.6 Electromagnetic-based Neuroimaging

EEG and MEG techniques directly measure the electric potential at the scalp and the magnetic field outside the head, respectively, produced by the neural activity in the brain. These methods are fully noninvasive and provide superior temporal resolution in the order of the neural activities, allowing monitoring of brain dynamics in the order of milliseconds. Unfortunately, the spatial resolution is limited by the spatial sampling (number of measuring electrodes), the effect of volume conduction on the signal, and the ill-posed nature of the electrostatic inverse problem (the inference of the current source generators of the measured EEG or MEG data) to one cm. Further, these methods are less sensitive to deep current sources in the brain.

As mentioned before, the origin of EEG and MEG signals is due to the primary and secondary postsynaptic currents. The primary (impressed) current flows in the dendrites from the synaptic activities to the soma, while the secondary (volume) current flows throughout the volume conductor [77]. The primary and secondary currents complete a closed current loop.

While both primary and secondary currents contribute to scalp potential and magnetic field, the main contribution to the scalp potential is from the volume current. For this reason, scalp potential is highly sensitive to volume-conduction properties such as tissue conductivities, structural variation of the skull [205, 149], conductivity anisotropy and inhomogeneity [125, 214]. Consequently, the electric potential suffers attenuation and spreading while propagating from the cortical sources to the scalp. For instance, the potential is reduced from the millivolts range at the cortex to the microvolt range at the scalp[34]. Therefore, it is possible that focal sources in the brain could be averaged in the scalp. This effect, called *blurring effect*, is the main drawback of the EEG technique. On the other hand, the main contribution to the magnetic field is from the primary currents. Therefore, the magnetic field is less sensitive to the volume-conduction properties–in particular, the low-resistive skull's propensity for allowing more sensitivity to deeper sources. Since we are more interested in the primary currents and the volume-conduction effect has less influence on the magnetic field, MEG becomes an attractive alternative technique.

Although, MEG overcomes some of the EEG limitations, it suffers from its own limitations. First, the method is insensitive to sources that are oriented normally to the scalp surface[88]. Therefore, MEG senses activities only from neurons in the fissures of the cortex. Second, the magnetic field outside the head is very small, in the order of femto Tesla (10^{-15} Tesla)[89], which requires sophisticated sensing technology (superconducting SQUIDs), an expensive operating team, and a less comfortable procedure for the subject, which limits-long term monitoring[89]. In contrast, typical EEG scalp potential is in the order of μV and thus can be measured using low-cost scalp electrodes and a high-gain amplifier, resulting in a comfortable procedure that allows long-term monitoring.

For these reasons, MEG doesn't provide major advantages over EEG. Nowadays, MEG and EEG are viewed as complementary rather than competing modalities. Several recent studies are focusing on combining the two methods[48], and most recent MEG research facilities are equipped with EEG as well.

3.7 High-resolution Neuroimaging

Electrical activation in the brain is a spatiotemporal process, which means its activity is three-dimensionally distributed and evolves with time. It is desirable to localize the brain's functional activities with high spatial and temporal resolutions. The temporal resolution of fMRI is limited to one second and can't be improved due to the physical constraint of a slow blood hemodynamic. On the other hand, EEG and MEG are the only available techniques that can measure the brain's functional activity with high temporal resolution. However, their spatial resolution remains on the order of a cm due to several factors, including [211]:

1. Head modeling errors
2. Source modeling errors
3. EEG measurement noise (instrumental or biological)

The standard is that spatial and temporal resolutions should be at least better than 5 *mm* and 5 *ms* [14]. Therefore, If the spatial resolution of EEG method can be improved to the order of *mm* by improving these factors, high-resolution neuroimaging can be achieved.

The brain response to stimulation is detected in terms of complex EEG data measured on the scalp. The problem of calculating the locations and orientations of the brain active sources that best explain the measured EEG data is called the *inverse problem*. This is a challenging ill-posed problem in neuroscience. A well-posed problem in neuroscience is the *forward problem*. The forward problem calculates what EEG or MEG datasets would look like for a set of source configurations. Accurate solutions of the EEG forward and inverse problems are required for high-resolution mapping of brain electric activity based on the EEG measurement technique. One of the main objectives of this dissertation is to improve the accuracy of the forward problem solution by improving the accuracy of the head model.

3.7.1 The EEG Forward Problem

The EEG forward problem determines the scalp potentials, given the current sources distribution in the brain and the head volume conduction properties. It is a well-posed problem and has a unique solution, governed by the quasi-static approximation of Maxwell's equations–Poisson's equation [153, 140]. Its solution defines the relationship between the neural electric sources (modeled as dipole current sources) and the scalp EEG electrode measurements. It is common to formulate this relationship mathmatically in terms of the so-called *transfer matrix* or *lead field matrix (LFM)* or *gain matrix (G)*.

The electric potential $\phi(\mathbf{r})$ at position $\mathbf{r}$ on the scalp, generated by a single dipole $\mathbf{q} = q\mathbf{e_q}$ with magnitude q and orientation $\mathbf{e_q}$ located at position $\mathbf{r_q}$ in the brain, can be obtained by solving Poisson's equation. Poisson's equation is linear with respect to the dipole moment (magnitude and orientation) and nonlinear with respect to its position. It is convenient to separate dipole magnitudes from their positions and orientations for reasons related to the inverse problem, as clarified in the discussion below. Therefore, the scalp potential can be written as, $\phi(r) = g(\mathbf{r}, \mathbf{r_q}, \mathbf{e_q})q$, where $g(\mathbf{r}, \mathbf{r_q}, \mathbf{e_q})$ is the potential at position $\mathbf{r}$ produced by a dipole with unit magnitude and orientation $\mathbf{e_q} = \mathbf{q}/q$, located at position $\mathbf{r_q}$. q is the magnitude of the dipole.

For M dipole sources, the electric potential at position $\mathbf{r}$ is the linear superposition of the contributions from each dipole,

$$\phi(\mathbf{r}) = \sum_{i}^{M} g(\mathbf{r}, \mathbf{r}_{qi}, \mathbf{e}_{qi})q_i,$$

where $\mathbf{r}_{qi}, \mathbf{e}_{qi}, q_i$ are the location, orientation and magnitude of dipole i, respectively. This expression can be rewritten in the form of a dot product of two vectors,

$$\phi(\mathbf{r}) = \mathbf{g}(\mathbf{r}, \mathbf{r}_{qi}, \mathbf{e}_{qi})\mathbf{S}^T,$$

where $\mathbf{g}$ is a M-vector containing the potentials generated by M-dipoles having unit magnitude, and $\mathbf{S}$ is a M-vector that contains the magnitudes of the dipoles. In this formulation, $\mathbf{g}$ describes the current flow from each dipole to the measuring electrode. Then, for N electrodes and M dipoles, the potentials at the electrodes generated by the dipoles in a matrix form are:

$$\mathbf{\Phi} = \mathbf{G}(\mathbf{r}, \mathbf{r}_{qi}, \mathbf{e}_{qi})\mathbf{S}^T,$$

where $\mathbf{G}$ is the gain matrix, which relates the set of M dipoles to the set of N sensors. Each column of $\mathbf{G}$ relates a dipole to the set of sensors called an *lead field* or *scalp topography.* This model corresponds to a single EEG time-sample. The model can be extended to include multiple time-samples when considering time-evolving activities in a straightforward way. In this case, $\mathbf{S}$ becomes a time-series matrix in which the time-series for each dipole is represented by the columns of $\mathbf{S}$, and the time-series for each electrode is represented by the columns of the time-series matrix $\mathbf{\Phi}$.

In this formulation, the forward model is called the “fixed” dipole model because the locations and orientations of the dipoles are fixed and assumed to be known *apriory.* The only varying parameters are the magnitudes of the dipoles. This is adequate, based on the fact that the dipoles that produce the measured field are oriented normally toward the cortical surface §3.3. In general, a noise matrix ϵ is added to the model and then the forward model becomes,

$$\mathbf{\Phi} = \mathbf{GS} + \epsilon. \tag{III.1}$$

The forward problem solution, based on realistic geometry and the use of the FDM-ADI algorithm, is the topic of Chapter IV.

3.7.2 The EEG Inverse Problem

The goal of the EEG inverse problem is to find an estimate for the locations, orientations and strength of the brain current source generators that can explain the measured EEG data. The problem is severely ill-posed for two reasons: (1) The solution is nonunique because there are an infinite number of source configurations that could explain a given EEG data set, and (2) the solution is unstable because it is highly sensitive to small amounts of noise in the measured data.

A brain current source generator is typically modeled as an electric current dipole uniquely specified by six parameters (three to specify the position and three to specificity the moment). However, the number of parameters can be reduced if some a priori constraints are placed on the sources as discussed below. Over the years, many methods have been formulated to overcome the ill-posed nature of the problem. These methods have been based on assumptions about the number of dipoles in the model and whether some of the dipole parameters are assumed to be known and remained constant. These methods can be categorized into two general approaches: the *parametric* approach and the *imaging* approach [48, 15, 82, 150]. The main difference between the two approaches amounts to whether a fixed number of dipoles are assumed a priori or not.

The Parametric Approach

The parametric approach, also called the equivalent dipole model, is based on the assumption that scalp EEG measurement is generated by one or a few current dipoles (less than 10) whose locations and moments (a total of six parameters for each dipole) are unknown. Then these parameters are estimated such that they produce scalp potential that best matches the measured EEG data. Considering the noise-free forward equation,

$$\mathbf{\Phi} = \mathbf{G}(\mathbf{r}_i, \mathbf{r}_{qi}, \mathbf{e}_{qi})\mathbf{S}^T,$$

the goal of the inverse problem is to find estimates for the sets $\mathbf{r}_{qi}, \mathbf{e}_{qi}$ and magnitudes, $\mathbf{S}$, that best explain the EEG data $\mathbf{\Phi}$. The easiest and most straightforward approach is to specify the number of dipoles M a priori, then minimize the difference between the EEG data and the scalp potentials computed from assumed source parameters by using the forward model. The difference measure between the computed and measured data sets, in terms of the least square distance, can be defined as,

$$E(\mathbf{r}_{qi}, \mathbf{e}_{qi}, S) = ||\mathbf{\Phi} - \mathbf{G}(\mathbf{r}, \mathbf{r}_{qi}, \mathbf{e}_{qi})\mathbf{S}^T||^2.$$

Then a nonlinear global minimization is carried out to ascertain the global minimum of the cost function $E(\mathbf{r}_{qi}, \mathbf{e}_{qi}, \mathbf{S})$ using a nonlinear search algorithm. The search for the global minimum starts with a specified set of parameters as a seed and proceeds iteratively. This involves solving the forward problem at each step. Different optimization methods can be applied to solve the nonlinear optimization problem–e.g., a multistart simplex search, genetic algorithms, or simulated annealing.

Various strategies can be applied based on the number of dipoles, which parameters need to be fixed, and whether to consider the time-series of the EEG data or just a snapshot. Several variations found in the literature include [165][49] (1) the single fixed dipole model, a variation in which the dipole location is fixed while its orientation and strength are variables; (2) the single moving dipole, in which the location and the moment are variables; (3) the single rotating dipole model, in which the location is fixed over a selected period and the orientation is allowed to vary within the period; (4) the multiple-dipole model, in which several dipoles are used to represent certain anatomical regions of the brain; and (5) the dynamic model, which takes into consideration the time-series of the dipole magnitude and the time-series of the scalp potentials. Further constraints on the dipole orientations, whether fixed or variable, can be added as well.

The major drawback of the least square method is that the number of dipoles must be specified a priori. Underestimating the number of dipoles causes the results to be biased by the missing dipoles. On the other hand, overestimating the number of dipoles causes the dipoles to fit any data, which makes it hard to differentiate the true dipoles; the solution becomes undetermined, incurring a performance penalty due to increasing the number of dimensions, and increasing the chances of the solution becoming trapped in local minima. Overcoming this outcome requires other approaches, such as beamforming methods. Several other parametric-based methods have been developed, including BESA, MUSIC, RAP-MUSIC, and FINES. A recent review about these techniques can be found in [82].

The Imaging Techniques

The imaging approach is also called the Distributed Source Model or Distributed Inverse Solution. In this approach the primary current sources are assumed to be current dipoles distributed to fixed locations within the brain volume. Since it is widely accepted that most EEG current source generators are restricted to the cortex, the dipoles are often restricted to the cortex with an orientation pointing normally toward the surface [15, 89]. Achieving high-resolution neuroimaging requires use of a very large number of dipoles (order of thousands or tens of thousands) to cover the large area of the convoluted cortical surface. Since the number of unknowns (number of dipoles) is in the order of thousands, while the number of sensors are only a few hundred (at most 256), the problem is severely underdetermined and thus requires imposing constraints on the allowed dipole distributions. Methods based on this approach apply imaging techniques to estimate the magnitudes of these dipoles such that the produced scalp potentials best explain the data. As formulated in section §3.7.1, the scalp electric potentials due to these dipoles and the current dipole magnitudes are related by the forward equation,

$$\mathbf{\Phi} = \mathbf{G}\mathbf{S} + \epsilon, \tag{III.2}$$

where, in the case of a single-time slice, $\mathbf{\Phi}$ is a column vector gathering the potentials at N scalp electrodes, $\mathbf{S}$ is a M-vector of the magnitudes of the cortical dipoles, ϵ is a perturbation noise vector, and G is a NM lead field matrix. Every row in $\mathbf{G}$ is a lead field corresponding to a current dipole (the solution of the forward problem at the scalp electrodes).

Given N-vector scalp EEG measurements $\mathbf{\Phi}$ at N electrodes and the lead field matrix $\mathbf{G}$ (computed a priori using dipoles with unit magnitudes), the goal of the inverse problem is to invert Equation III.2 and find an estimated solution of the dipole moments magnitudes, $\mathbf{S}$.

Since the only variables are the magnitudes of the dipoles **S**, and the scalp potential is linear with respect to them, the problem is linear and the inverse problem is reduced to find a solution of a linear inverse problem for unknown magnitudes (vector **S**). This is a well-known formulation for numerous image reconstruction problems. Methods for solving this problem take regularizing schemes into account to overcome the ill-posed nature of the problem. Various methods and variations are developed based on this technique, including minimum norm estimates and their generalizations, LORETA, sLORETA, eLORETA, VARETA, S-MAP, ST-MAP, Backus-Gilbert, LAURA, Shrinking LORETA, FOCUSS (SLF), SSLOFO, and ALF. A survey of these methods and comparisons between some of them can be found in [82, 15, 48]. Of these methods, LORETA is one of the most widely accepted. From the formulation of the inverse problem, we see that no matter how sophisticated the inverse technique, the accuracy of the inverse problem depends on, and is limited by, the accuracy of the forward solution. Part of this dissertation's focus is to provide sophisticated HPC computing tools and methods to improve the solution of the forward problem–in particular, the conductivity model of the head tissues, especially the highly resistive skull.

3.8 Summary and Conclusions

Neurons are the basic elements of information processing in the human brain. Active neurons are electrically and metabolically active. They generate electric and magnetic fields and consume energy and oxygen more than inactive neurons. Achieving high-resolution neuroimaging requires that the locations of brain active regions be determined in high spatial and temporal resolutions (order of mm and ms). Two kinds of techniques are used to accomplish this.

1. Direct techniques (EEG/MEG) are based on measuring the electric and magnetic fields associated with the active brain regions by using sensors directly on the scalp. Then the source generators of the fields are inferred from the measurements.

2. Indirect techniques (PET/fMRI) are based on inferring the brain active regions by measuring the metabolic and hemodynamic properties of the brain.

Indirect techniques provide high spatial resolution (order of mm), but suffer from low temporal resolution (minutes in the case of PET and a second in the case of fMRI), which can't be improved due to the physical constraints of slow metabolic and hemodynamic responses. On the other hand, direct techniques provide excellent temporal resolution (order of ms), but the spatial resolution is poor (order of cm) due to head-modeling errors and source-modeling errors, as well as sampling and noise in EEG measurements.

The advantages of EEG are its superior time resolution, noninvasiveness, low cost, and comfort for subjects, which allows long-term monitoring; therefore, EEG can provide a unique window on the dynamics of brain functions if spatial resolution is improved. Spatial resolution can be improved by improving the accuracy of the aforementioned sources of error. Current advances in EEG technology allow spatial sampling of up to 256 electrodes, which improves the interelectrode distance to about 1.25 cm. Applying modern signal processing methods increases the accuracy of solving the inverse problem and source modeling.

However, no matter how sophisticated the inverse procedure, the accuracy of source localization will be limited by the accuracy of the forward solution. The accuracy of the forward solution depends on the accuracy of the volume conduction model and the numerical method. Up until recently the human head was approximated as a three- or four-shell spherical model where each shell corresponds to a specific head tissue (e.g., brain, skull and scalp). Advances in structural imaging techniques such as MRI and CT provide accurate geometry that defines the boundaries of the head tissues with accuracy better than $1mm$; consequently, forward solutions based on realistic geometry have recently become common.

In addition to improvements in the geometrical model, an accurate conductivity model of the head tissues–especially for the skull, which is most resistive to measurements–has become equally or more important. Until now, lack of accurate knowledge about head-tissue conductivities, especially the highly resistive skull, has been the main source of errors in head modeling. Skull conductivity is particularly problematic given the developmental variations in the skull from infancy through adolescence and the variations across subjects. Without an accurate model of the skull, even advanced inverse efforts cannot achieve precision with EEG data, as the error of source localization due to conductivity uncertainty may reach a few centimeters [96].

One major goal of this dissertation is to provide tools and methods that can be used to improve the spatial resolution of EEG-based source localization. In particular, our focus is to improve the forward calculation that predicts the scalp potentials associated with a specific brain active region. We accomplished this by providing two methods. The first method is a technique to estimate the conductivities of the tissues associated with the subject's realistic geometry obtained from MRI or CT images. To accomplish this, we solved what is called the conductivity inverse problem or the bounded Electrical Impedance Tomography (bEIT) V. The second method is an improvement in the skull model that involves including skull inhomogeneity in the forward calculation IV. In addition to these methods, we provided methods of analysis to investigate and quantify the effect of the model input parameters. Further, we factored out these methods into a general purpose extensible environment for scientific investigations. This environment can be used to conduct similar investigations in other domains or on a different model within the domain.

CHAPTER IV

HUMAN HEAD ELECTROMAGNETICS

Fundamental problems in neuroscience involving experimental modalities like EEG and MEG are naturally expressed as tomographic imaging problems. The difficult problems of source localization and impedance imaging require modeling and simulating the associated bioelectric fields. The source localization inverse problem involves estimation of the current sources in the brain that generate EEG/MEG signals. The conductivity inverse problem involves the estimation of the head tissues' conductivities that explains measured data induced by known currents injected from the scalp. Before such estimations can be made, we must solve the forward problem. The forward problem is well-posed and has a unique solution governed by Poisson's equation, the quasi-static approximation of Maxwell's equations. Solving the forward problem starts from a given set of current source configuration, the geometry of the head tissues and their conductivities. Then the potentials at the scalp electrodes and the magnetic field can be calculated by solving Poisson's equation. Since solving either of the inverse problems involves a large number of forward calculations, it is important that the computational method of the forward problem is accurate, stable, and fast.

4.1 Maxwell's Equations

Assuming $\mathbf{\Omega}$ is a volume conductor with an arbitrary shape under investigation, $\Gamma_{\mathbf{\Omega}}$, is the boundary of the volume conductor. The current density within the volume induces electric and magnetic fields **E** and **B** that can be measured on the surface of the conductor. Assuming the conductivities σ and the electrical current sources S are known, the electric and magnetic fields inside the volume are fully described by Maxwell's equations,

$$\nabla \times \mathbf{H} = \mathbf{J} + \partial_t \mathbf{D}, \tag{IV.1}$$

$$\nabla \times \mathbf{E} = -\partial_t \mathbf{B}, \tag{IV.2}$$

$$\nabla \cdot \mathbf{D} = \rho, \tag{IV.3}$$

$$\nabla \cdot \mathbf{B} = 0, \tag{IV.4}$$

and the continuity equation,

$$\nabla \cdot \mathbf{J} = -\partial_t \rho, \tag{IV.5}$$

where **E** and **H** are the electric and magnetic fields, **D** is the electric displacement, **B** is the magnetic induction, ρ, and **J** are the electric charge and current densities.

Since biological tissues behave as electrolytes ([153]), the relations between the fields are,

$$\mathbf{D} = \epsilon\mathbf{E}, \tag{IV.6}$$

$$\mathbf{B} = \mu\mathbf{H}, \tag{IV.7}$$

$$\mathbf{J} = \sigma\mathbf{E}, \tag{IV.8}$$

where the parameters ϵ, μ and σ denote permitivity, permeability and conductivity of the medium. In general, these parameters are tensors for the anisotropic medium and scalar for the isotropic medium.

4.2 Quasi-static Approximation

In the case of EEG/MEG and the properties of human head tissues, it is possible to simplify Maxwell's equations in a form that is easier to solve. In the first approximation, the time-dependent terms can be ignored since the relevant frequency spectrum in EEG and MEG signals obtained from measurement is typically below $1kH_z$ and most studies deal with frequencies between 0.1 and $100H_z$ [89]. In the second approximation, the capacitive component of the tissues' impedance and the inductive effect can be ignored because the conductivities σ of the tissues are in the range of $2.0S/m$ for CSF tissues to $.001S/m$ for the skull [194] [153] [89]. If we ignore the time-dependent terms in equations IV.1 and IV.2 and assume the magnetic permeability μ is constant and equal to the vacuum permeability, Maxwell's equations become,

$$\nabla \times \mathbf{H} = \mathbf{J}, \tag{IV.9}$$

$$\nabla \times \mathbf{E} = 0. \tag{IV.10}$$

Equation IV.10 implies that there is a scalar field V (electric potential) such that $\mathbf{E}$ is the negative gradient of V,

$$\mathbf{E} = -\nabla V. \tag{IV.11}$$

Taking the divergence of equation IV.9, we get,

$$\nabla \cdot \mathbf{J} = \nabla \cdot \nabla \times \mathbf{H} = 0, \tag{IV.12}$$

since the divergence of the curl of any vector field is zero. The total current density $\mathbf{J}$ inside the volume is the sum of the primary (impressed) current $\mathbf{J}_s$ and the volume ohomic (or return) current $\mathbf{J}_\Omega = \sigma\mathbf{E}$,

$$\mathbf{J} = \mathbf{J}_s + \mathbf{J}_\Omega = \mathbf{J}_s + \sigma\mathbf{E}. \tag{IV.13}$$

The primary current is the source current obtained from the neural activity. The volume current is the ohmic current resulting from the effect of the electric field on moving the charges in the volume conductor. σ denotes a second-rank tensor.

Taking the divergence of IV.13 and using equation IV.12, we get,

$$\nabla \cdot (\sigma \mathbf{E} + \mathbf{J}_s) = \nabla \cdot \mathbf{J} = 0. \tag{IV.14}$$

Finally, if we use $\mathbf{E} = -\nabla V$ in equation IV.14 and assume Neumann boundary conditions, since no current flows out of the head, it follows that the electric potentials solve the following boundary value problem,

$$\begin{aligned} \nabla \cdot (\sigma \nabla V) &= \nabla \cdot \mathbf{J}_s, \quad in\ \mathbf{\Omega}, \\ \mathbf{n} \cdot \sigma \nabla V &= 0, \quad on\ \Gamma_{\mathbf{\Omega}}. \end{aligned} \tag{IV.15}$$

Equation IV.15 is Poisson's equation in 3D. It defines the relationship between a current source and the potential at any position within the volume. In a similar way, expressions for the quasi-static magnetic field can be obtained. But in this dissertation we consider only EEG data.

In this approximation, the potentials generated by a given current source depend only on two things: (1) the magnitude, location and orientation of the current sources; and (2) the characteristics of the volume conductor, such as the geometry of the tissues and their electrical properties. In terms of these things, the forward problem can be stated as follows: given the positions, orientations and magnitudes of current sources, as well as geometry and electrical conductivities of the head volume Ω, calculate the distribution of the electrical potentials V on the surface of the head (Scalp) that satisfy the boundary conditions.

In finding the solution of the source localization problem, one performs many forward calculations for different current source configurations until an optimal configuration is found that the calculated and measured EEG data best match. Therefore, the accuracy of the inverse problem is highly dependent on the accuracy of the forward problem, and the accuracy of the forward problem depends on the accuracy of the volume conductor characteristics.

4.3 Volume Conductor Models

When the electric potential propagates from current sources in the brain, through the head tissues to the measuring sensors, it is affected by the medium. This effect, called *volume conduction effect*, is the main source of inaccuracy of the forward solution. To improve the forward solution accuracy, one must define and model the volume conductor characteristics (the electrical conduction properties). The main characteristics are the geometry that defines the boundaries between the tissues and the conductive properties of the tissues such as inhomogeneity and anisotropy. A complete, fully realistic volume conductor model is not expected in the meantime. However, specification of the main characteristics and quantification of their effect on the forward solution are important factors in improving the solution accuracy.

Several models have been developed to solve the forward problem. In these models, the human head consists of multiple tissues–e.g., brain, skull, CSF, and scalp. Even though the forward problem is well-posed, an exact analytical solution is available only for simple geometries such as multishell spherical models. Solution for complex realistic geometries is only available numerically and its accuracy is limited by the characteristics of the volume conductor, which are either poorly known or too difficult to be modeled. For this reason, simplifications of one or more properties are necessary. These simplifications usually involve the geometrical model of head tissues and their conductive properties.

Improving the forward solution and consequently the inverse solution depends significantly on improving these simplifications. Different methods enforce different levels of simplification. For example, it is not possible to include tissue anisotropy or inhomogeneity in the widely used BEM solver or multishell spherical model.

In the following sections, we review the literature on the volume conductor description of the head. This includes evaluation of the effect of geometrical and conductivity models of the main head tissues on the forward and inverse problems.

4.3.1 Spherical Models

The simplest possible volume conduction model of the human head consists of a single homogeneous sphere [64]. However, this model assumes that the electric properties of all head tissues are uniform and isotropic, which is far from reality given the significant difference between skull conductivity and other tissues (a factor of 20). As a first improvement to this model, multishell models are introduced. In these models the head is approximated as concentric spherical shells representing tissues with homogeneous isotropic properties. The most widely used multishell model is the three-shell spherical model representing the three major head tissues, brain, skull and scalp [169], [171] [42] [22]. It has been studied experimentally using a skull soaked in saline and the results show good qualitative agreement with a variety of general observations of EEG data [140, 221]. For further improvement, models that include CSF tissue in a four-shell spherical model become common as well [61], [43] and five-shell models are also investigated [203].

These models capture the major head tissue layers, and their simple geometry allows solving Poisson's equation analytically [11] [184]. Solving the forward problem becomes an evaluation of a semi-analytic function, which involves the computation of a truncated infinite series. For further efficiency improvement, other variations are introduced such as using a single-spherical model that approximates the multishell spherical models [22]. Also, to address conductivity anisotropy in these models, one can use a semi-analytic solution that allows conductivity in the tangential direction to be different from conductivity in the radial direction [51][50].

However, these models have obvious limitations. The human head is not spherical and the head tissues do not have uniform thickness and conductivities [45]

[203, 204]. For instance, the brain white matter and the skull conductivies are inhomogeneous and anisotropic [215]. Further, the skull contains holes and other variations that have a significant impact on the accuracy of the solution [44, 103, 33], in which a hole in the skull causes an error of up to 2 cm [21]. For example, an error of 10-20 % [41] [167, 40] in the scalp potentials and 1-2 *cm* in source localization [167, 168, 102, 225, 147] can be obtained due to the geometry factor alone. Further, it was found that using a realistic model, the error in source localization is reduced from 2-3 cm to 1 cm [53] when using spherical models. Another study using a realistic head-shaped model indicated that the sphere model is not accurate for computing the magnetic fields corresponding to deep sources or sources in the bottom region of the skull [89][186]. Including these important characteristics in multishell spherical models is not possible.

4.3.2 Realistic Models

Structural imaging modalities such as magnetic resonance imaging (MRI) and computational tomography (CT) can provide images that define the boundaries between the head tissues with resolution better than 1 *mm*. Forward models based on geometrical information obtained from these images are called *realistic head models* [103, 225]. They have recently become standard, as they capture accurate geometry of the head and allow the inclusion of various conductive properties of the tissues such as anisotropy and inhomogeneity. These models provide better forward solution accuracy compared to spherical models. However, their computational complexity and requirement become higher as well.

To deal with the complex geometries, PDE solvers use Boundary Element (BE) [88] [18, 69], Finite Element (FE) [200] [12, 204], or finite difference (FD) methods [107, 199, 89, 21]. The main computational idea behind these methods is to reduce a continuous problem with infinitely many unknown field values to a finite number of unknowns. This is achieved by discretizing the solution region into elements. Then the conductivity values are assigned to each element.

Application of these approximation methods to the governing equations for the specific modality eventually yields a system of linear equations of the form $AX = b$, which must be solved to obtain the potential at the scalp X, where b corresponds to the electrically active sources and A is the stiffness matrix. The solution techniques can be broadly categorized as direct and iterative solvers. The choice of a particular solution method is highly dependent upon the approximation technique employed to obtain the linear system, the size of the resulting system, and the accessibility of computational resources [5].

Boundary Element Method

The Boundary Element Method (BEM) [88] is the earliest realistic head model. It is widely used due to its computational performance. The idea behind BEM is the transformation of the volume partial differential equation to an integral equation over the surfaces that bound the volumetric tissue compartments, using Green's theorem [17, 198, 133]. This formalism reduces the dimensionality by one dimension, which increases the method performance. BEM uses geometrical information from structural images to define the interfaces between different tissue types within the volume. Typically, in head modeling, nested volume topology is used and the volume conductor consists of three surfaces: brain-skull, skull-scalp and scalp-air. The regions between the surfaces are assumed to be isotropic and homogeneous. Then the given boundary conditions are used to fit the interface value into the integral equations. When the surfaces are digitized into triangles the integrals are turned into sums. Then the solution is obtained by solving the resulting dense linear system of equations.

The main attractive feature of BEM is its low level of computational needs in finding the potential at the scalp surface. Since the problem is 2D, the number of elements is relatively small and direct inversion techniques are used. After the matrix inversion, only matrix multiplication is required to compute the potentials for a different source configuration. However, the main problem with BEM is that it is restricted to the case of homogeneous and isotropic conductivities and does not allow including other structural variations such as skull holes [126]. Another drawback of BEM is that the resulting linear system of equations is dense. This means if a large number of elements (triangles) is used to improve the accuracy [69], the memory requirement becomes high. However, advances in memory technology overcome this limitation. Further, a variation of the method for repeated computation of the forward solution in the computation of the inverse problem using interpolation can be as efficient as spherical models [59]. The BEM limitations motivate the use of Finite Element and Finite difference methods in head modeling mainly to include skull inhomogeneity and anisotropy.

Finite Element Method

In the finite element method (FEM), the entire volume conductor is descreteized into small elements[12, 27]. The elements can have various volumetric shapes such as a tetrahedron or cube. Different properties can be assigned to these elements, which allows including complex characteristics of the volume conductor such as conductivity anisotropy [125, 217, 91] and inhomogeneity, and also allows including other structural variations such as holes [92]. Including these characteristics improves the accuracy of the forward solution. However, there is a performance penalty for this flexibility since FEM digitizes the 3D space instead of 2D in the case of BEM, the number of elements is much higher, and the size of the stiffness matrix becomes larger. Therefore, direct inversion techniques are not possible and an iterative procedure is required to invert the matrix. Improving the performance requires the use of various approaches to domain decomposition, which produces elements with variable sizes. Another problem of FEM is that mesh generation for a 3D highly heterogeneous segmented image with irregular boundaries (e.g., the human head) is a difficult task. The process involves a significant degree of preprocessing and smoothing of the initial geometry through manual means, which introduces another potential source of error. A fully automated process of image segmentation and mesh generation is unavailable at present. Another important consideration is how to model a dipole source in FEM [187]. In case of a point current dipole the singularity of the potential at the source position is treated with a "subtraction dipole model" in which the total potential is divided into the analytically known singularity potential (a dipole in an infinite homogeneous domain) and the singularity free potential, which then can be approximated numerically [187] [23] [206][216, 217]. To address these difficulties, the Finite difference method provides a simpler approach, but with higher computational requirements.

Finite Difference Method

The FDM method is based on replacing the derivatives in PDE by finite differences. In FDM head modeling, the head is tessellated in a regular cubic grid, where each cube has the same size. Using the FDM method with a regular cubed grid is generally the easiest method to code and implement. It is often chosen over FEM methods for simplicity and the fact that an MRI/CT segmentation map is also based on a cubed lattice of nodes [119, 208, 131, 136]. Therefore, meshes are easy to construct (once segmentation is accomplished) as the cubic/rectangular elements can be “mapped” directly from the voxels of the medical images (3D MRI scans). Many anatomical details (e.g., olfactory perforations and the internal auditory meatus) or structural defects resulting from trauma (e.g., skull cracks and punctures) can be included as the computational load is based on the number of elements and not on the specifics of tissue differentiation. Thus, the model of geometry accuracy can be the same as the resolution of MRI scans (e.g., $1 \times 1 \times 1mm$). In contrast, in the FEM approach, simplification of the geometry is unavoidable as a result of mesh generation, which can be too difficult and introduce errors.

The number of forward calculations for different source configurations can be reduced significantly by using the reciprocity principle [87]. In a study by [131], the researchers conduct a performance study of several FDM-based methods, successive over relaxation, preconditioned conjugate gradients and an algebraic multigrid for solving the EEG forward problem. They found that an algebraic multigrid is computationally the most efficient. Similar to FEM, FDM allow incorporating anisotropy of the tissue [87] [173, 174].

Summary and Discussion

Realistic head models have recently become standard, as they provide major improvement over the spherical models. BEM is based on solving the surface integral equation instead of the volume partial equation, which reduces the dimensionality to 2D. This makes it efficient in terms of computational requirements. However, they are restricted to the only case of an electrically homogeneous, isotropic and closed tissue compartment. In contrast, FEM and FDM are based on digitizing the whole volume conductor into small volumetric elements, which allows assigning different a conductivity value for each element and consequently including various modeling properties such as inhomogeneity and anisotropy. The price of this flexibility is performance. In FDM and FEM the stiffness matrix is much larger and can't be solved by a direct method. Typically, iterative techniques such as successive over relaxation, conjugate gradients and algebraic multigrid are used [131, 97]. The drawback of the iterative techniques is that the solver has to be reapplied for each source configuration. Therefore, FEM and FDM are computationally inefficient in solving the inverse problem in which an iterative solver needs to be used for each dipole. However, using the reciprocity theorem overcomes this inefficiency [170, 87].

Since in FDM, the 3D space is a regular cubed space, where the computational points lie fixed, while in FEM the computational points (the vertices of the elements) can be chosen freely, FEM requires much fewer points to represent the irregular interface between the different tissue compartments. This makes FEM more efficient than FDM, as the number of elements is much smaller (.5M compared to 5M). However, constructing an FEM mesh from the segmented high resolution MRI image is a difficult task. It can be inaccurate due to the difficulty in differentiating the boundaries between different tissues in a structure with complex geometry such as the head. On the other hand, in FDM the regular cubed images map directly to the computational grid without any effort. Typically, preconditioning techniques are used to improve performance in solving the linear system [220].

For these reasons our choice was to use the FDM-based alternating direction implicit (ADI) method. The method is implicit since it requires linear systems solutions and alternates since it alternates on the three directions (x, y, z). We chose the ADI method because solving the system requires a number of uncoupled tridiagonal solutions that require little memory and can be executed concurrently on different processors.

4.3.3 Conductivity Model

The use of realistic geometries, obtained from structural imaging such as CT and MRI, is now standard in the forward calculation. However, this knowledge alone is not sufficient to describe the volume conductor accurately because EEG signals are highly sensitive to tissues' conductive properties [92, 90, 125, 147, 205, 127, 154]. This sensitivity is due to the volume current (Ohmic current, $J = \sigma E$), which is the main source of EEG signals. The volume current is highly dependent on the characteristics of the medium. Therefore, accurate knowledge about the electrical properties of head tissues is equally or more important than the geometrical information [140]. The regional conductivities of the tissue compartments are largely unknown or poorly known, especially for the skull, given their development variations within a subject and their variations across subjects. In contrast, MEG is insensitive to the tissues' conductivities since the sources of MEG signals are the primary currents [146]. Multishell spheres and BEM models inherently assume each tissue conductivity is uniform and isotropic (anisotropic spherical models exist that allow different radial and tangential conductivities), ignoring conductivity variation within the tissue. Hence, in these models only the ratio between the conductivities of adjacent compartments is required. This is why most of the early work was focused to obtain the ratio of the conductivities instead of the absolute values, as discussed below. On the other hand, knowledge about absolute conductivity values can be included in FDM and FEM models, which allow modeling inhomogeneity and anisotropy and provide more accurate volume conductor description.

In general, the conductivity of a biological tissue is related to the body fluid concentration level in its content [115]. Tissues with higher concentration of fluid are more conductive because body fluid is rich in salt and so in ions, which are the carriers of the current. Cell-free fluids such as urine and CSF have the highest conductivity [140], while compact bones have the lowest conductivity.

In the following subsections, we discuss the electrical properties of the head tissues in general, focusing on the skull.

Brain

The brain consists of neurons and glial cells. Neurons are the basic information-processing cells. Their cell bodies and dendrites are located in the gray matter, while their axons extend into the white matter. Most of the gray matter is located in the cerebral cortex, which forms the upper layer of the brain. The inner part of the brain consists of white matter. White matter consists of nerve fibers (bundles of axons) that connect different parts of the cortex together, and the cortex with other peripheral parts. Since ions can move easier along a fiber nerve than perpendicular to the nerve, the conductivity in the direction along a nerve is higher than in the direction perpendicular to the nerve. Measurements show that the ratio between the conductivity along the nerves to the conductivity across the nerves is 9:1 [138, 84]. Therefore, the conductivity of white matter is anisotropic and should be included in the volume conductor for better accuracy [91, 125, 217, 85, 87, 158, 84]. On the other hand, gray matter is homogeneous and isotropic tissue.

The recent Diffusion Tensor Magnetic Resonance Imaging technique (DT-MRI)[19] provides directional information about the diffusion tensor of water. Since ions move easier with the water flow, it is assumed that the conductivity is higher in the flow of water direction. Therefore, it is possible to infer the conductivity tensor from the water diffusion tensor [202].

Some experimental data about the conductivities of white matter and gray matter for humans and animals are available. Latikka et. al. [117] measured the resistivity of white matter, gray matter and CSF obtained from nine tumor patients (6 males and 3 females ranging in age from 32 to 87) during surgery. Their results show the average resistivity of white matter is 3.91 ohm m (conductivity of .256 S/m), the average resistivity of gray matter is 3.51 (conductivity of .284 S/m) and the average resistivity of tumor depends on the type of tumor and range from 2.3 to 9.7 ohms m (.43 to .1 S/m).

In many head modeling systems the brain is modeled as a single tissue in the popular three-layer models (scalp, skull, brain). However, this approximation is inadequate. The conductivity of the brain is inhomogeneous since the gray matter is about twice as conductive as the white matter, and the conductivity of the white matter is anisotropic. Spherical and BEM models inherently ignore brain anisotropy and inhomogeneity, while FDM and FEM models allow including anisotropic information and inhomogeneity in the forward calculation.

CSF

Cerebrospinal fluid (CSF) is a colorless liquid that fills out the space around and within the brain and throughout the nervous system. Its main function is to provide support and protection. It acts as a shock absorber for the brain and provides other support functions such as circulating nutrients and removing waste from the nervous system.

Since CSF is a body fluid, its conductivity is expected to be homogeneous, isotropic, and higher than other body tissues. Also, it is expected to decrease with age as it gets dryer. Baumann et. al. [20] measured the conductivity of CSF taken from 7 patients across a frequency range 10 Hz–10kHz. The measured conductivity was 1.45 S/m at room temperature and 1.79 at body temperature. Currently, this is the typical value used for the conductivity of CSF tissue in head modeling. In another study, Latika et al. [117] measured the conductivity of CSF from two patients with brain tumors during surgery at frequency 50kHz. Their results show an average CSF conductivity of 1.25 S/M, lower than Baumann's results. In head modeling the scalp potential is insensitive to CSF conductivity, as it doesn't vary and typically is fixed.

Skull

The low skull conductivity, compared to other tissues, makes the scalp potentials highly determined by this parameter.

Until now the skull conductivity has been poorly known and the object of research in simulation and experiment. Published data are not consistent and cover a wide range of values. The lack of accurate skull conductivity estimates is particularly problematic, given the developmental variations in the human skull from infancy through adolescence. Without an accurate skull model, even advanced inverse efforts cannot achieve precision with EEG data. Current advances in EEG dense array electrode technology, the availability of up 256 electrodes, and the accurate geometrical construction of head tissue make the accuracy of the forward solution (and therefore the inverse problem) mostly limited by the conductivity modeling of the skull. Symmetric forward models such as multishell spheres and BEM require only the ratio of the conductivities scalp:skull:brain to be correct. That's why most of the head-modeling studies in past decades were focused on getting the skull:brain ratio correct. However, in FEM and FDM, more structure can be modeled, such as inhomogeneity and anisotropy; therefore, absolute conductivity values of each tissue are necessary.

Several sensitivity studies were conducted to evaluate the influence of skull model parameters on the accuracy of the forward and inverse solutions. The outcome of these studies is that inhomogeneity, anisotropy, and other skull variations, such as holes, must be included in the skull model. Using the FEM model, Pohlmeier [154] studied the effect of skull conductivity variations on source localization and showed that it is necessary to include the skull three-layer model and the local variations of skull conductivity. In their study, conductivities of the lower and upper layers of the skull were assumed to be equal. However, another study [33] using spherical three-dimensional resistor mesh indicated that differences resulting from using three different layers for the skull instead of 1 layer are small, while introducing a hole in the skull causes changes in the potential by a factor of 11.5 and the largest error in the source localization. Benar et. al. [21] obtained similar results. When they introduced a hole in the skull they found a localization error up to 2 cm for radial dipole and a reduction in the source localization error by a factor of 10-20% when the skull conductivity was doubled. Using the FEM model, Marin et al. [125] investigated the influence of skull anisotropy on the forward and inverse problems. Their results show that the conductivity of the skull in the radial orientation has more effect on the forward and inverse solution than the tangential direction. Using the FEM model, Wolters et al. [215] found that skull anisotropy has a smearing effect on the forward solution and has more effect on the potentials generated by deeper sources. Using an FEM model, Ollikainen et al. [147] studied the effect of skull inhomogeneity on the source localization. Their results indicate that source localization errors of about 1 cm can be encountered if the inhomogeneity of the skull conductivity is not considered. In [92, 90], they studied the influence of conductivity changes on the magnetic field and the electric scalp potential. Their results indicate that the magnetic field is insensitive to variation in conductivity, while the scalp potential is highly sensitive to tissue conductivities (both dipole localization and dipole strength). Using a 3-shell spherical model, Eshel [60] conducted a correlation study between matching skull thickness asymmetry and scalp potentials. Their results show that skull thickness asymmetry can create nonnegligible asymmetries in the potential measured on the scalp above homotopic points of the two hemispheres.

Anatomy and structure. The human skull consists of 8 cranial bones and 14 facial bones joined together by sutures. Skull bones can be classified into two kinds according to their material: compact (dense) bones and spongy (cancelous) bones. In general, the local conductivity of a bone is highly related to the concentration of fluids in the bone [115]. Regions with higher fluid concentration are expected to be more conductive than regions with lower concentration. Sutures are composed of materials that are highly rich with fluids. Spongy bones contain higher concentration of fluids (marrow) compared to compact bones. Therefore, sutures are expected to be highly conductive, and the conductivity of a spongy bone is expected to be much higher than the conductivity of a compact bone. Even more, the conductivity of a composite bone (a bone made of a combination of spongy and compact bones) is expected to be highly dependent on the fraction of the spongy bones in the composite. Indeed, these general observations are confirmed by experimental measurements of the conductivity of live and dead skulls[118, 9, 8, 197].

The cortical part of the skull consists of three-layer bones, called *trilayer bones.* A trilayer bone is formed of a spongy middle layer (thickness 3.8–5.1 mm) sandwiched between two compact-bone layers (thickness 1.7–4.3 mm). The concentration of fluids in the lower compact bone layer is higher than in the upper compact bone layer, so the lower layer is expected to be more conductive than the upper layer and the middle layer to be much more conductive than the outer layers. This is confirmed by an experimental study on a live and dead skull [9], where the conductivity of each layer is measured separately. A long bone is generally a composite bone formed from spongy inner bones surrounded with a shell of compact bones. The thickness at the middle of a long bone is mostly made of compact bones with a small amount of spongy bones; while at the end of the bone, it is mostly made of spongy bones with a narrow shell of compact bones.

In addition to variations in bone type, structural variations within the skull such as openings and thin regions have a large impact on the effective conductivity of the whole skull. These holes and openings are filled with nerves and body fluids, which provide the current easier paths to pass through the skull and consequently increase the effective conductivity of the whole skull. The structural variations effect becomes significantly important in infants and young children, where the skull is not completely developed [63]. Using a realistic FEM model, Jing [108] studied the effect of holes on EEG data. His results show that the strongest effect on EEG occurs when the dipole is located below the center of the hole for radial dipoles and when the dipole location is just below the border of the hole when the dipole is tangential. Due to these structural and anatomical variations, it is expected that skull conductivity is highly inhomogeneous and anisotropic.

Experiment data. Experimental data show considerable variation in skull conductivity. Gabriel [70, 71, 72] summarizes the results of earlier tissues' dielectric properties. In a study on samples obtained from a dead skull, law [118] found that compact bones have the lowest conductivities, sutures have the highest conductivities, and the conductivity of trilayer bones is linearly proportional to their thickness. The study suggests that trilayer bones' conductivity might be determined by their thickness. The conductivity of a uniform material is not expected to depend on its geometry. But in the case of a composite material, such as a trilayer skull, it is possible to be related. This can be explained because if the thickness of the spongy middle layer bone increases proportionally more than that of the compact layers in the thicker part of the skull, the conductivity of the thicker region is expected to be higher than that of the thinner samples. These general observations are confirmed in a more recent study [197].

Later, Akhtari [9] measured the conductivity of individual skull layers and the bulk skull at several locations, using the live skulls of four subjects. Their results show that the three layers of the skull have different conductivities: the inner layer conductivity is higher than the outer layer conductivity, and both have lower conductivity than the middle spongy layer by a factor of three to six. Further, all conductivity values of the skull are lower than what has been previously published, and the conductivity of each layer is not uniform. Also, the results show that conductivity of the bulk skull has a weak dependence on thickness. However, the study has only a few points (3–4), which is insufficient for drawing conclusions.

A recent intensive study on 388 live skull samples by Tang [197] indicates a strong dependence of the conductivity of a sample on the fraction of spongy bones in its content. Also, these results confirm the linearity relationship, obtained earlier by Law, between conductivity and thickness for trilayer bones. Similar to Law's results, Tang's results show that sutures significantly increase local conductivity. Further, the conductivities of the suture in their results are higher than those obtained by Law. This is explained due to the fluid content of their live samples compared to the saline content of a dead skull.

Until recently most of the brain research assumed a brain-to-skull conductivity ratio of 80 based on measurement obtained by Rush and Driscoll [169, 171] on a dead skull hydrated with saline. Since dry skull is effectively an insulator, the effective conductivity of the skull is proportional to the conductivity of the fluid with which it is permeated. Therefore, Rush and Driscoll found that the conductivity ratio of the permeating fluid (saline) to the soaked skull was 80. This number became the standard parameter in the forward computation for decades. In 1983 the same results were suggested by Cohen [38] in a combined analysis of the (EEG) and(MEG) recordings evoked by the same stimulus.

Completely different results were obtained later in two studies by Oostendorp [148]. One study was in-vitro on a dead skull and the other one was in-vivo on two subjects using scalp current injection and electrodes at 32 positions. In the in-vivo study, they used scalp current injection and BEM for the forward calculation. Then they constrained skull conductivity as a fraction of brain conductivity and fit the data to only one parameter to find the conductivity of the skull. Their result indicates a brain-to-skull conductivity ratio of 12-20. In their in-vitro study on a dead skull soaked in saline, the average conductivity of the skull was .015 S/m at frequency range 100Hz to 10kHz.

Hoekema [96] questioned the validity of the previous studies on skull conductivities due to exposure to the air (even for a few minutes) and lack of temperature control. Avoiding these drawbacks, they measured the mean conductivity of 5 live skull parts that were temporarily removed during epilepsy surgery at body temperature and high humidity conditions using 32 electrodes. A finite difference model was used to compute the EIT inverse problem. Their results show higher conductivity than previous results of value ranges from .032 to .08.

Using scalp current injection, Goncalves [79, 80] conducted an in-vivo study on 6 subjects. They obtained a wide range of brain-to-skull conductivity ratios. Their results were model dependent. When they used a 3-shell spherical model in the forward calculations, the average brain-to-skull conductivity was 72. However, when they used realistic geometry and BEM, the ratio was 20-50.

In a study on 5 epilepsy patient children, Lai [116] found the effective brain-to-skull conductivity ratio in the range of 18–34 with an average of 25.5. In their study, they simultaneously recorded scalp and cortical electrical potentials during subdural electrical stimulation. The inverse cortical potentials distribution was then computed from the scalp-recorded potentials using a 3-shell spherical model. Then the brain-to-skull conductivity ratio was estimated by minimizing the difference between the recorded cortical potentials and the computed cortical potentials.

In a similar study by Zhang [227] on 2 children, both epilepsy patients, they found a brain-to-skull conductivity ratio of 18. In this study, intra-cranial electrical stimulation was delivered using a pair of electrodes in the implanted subdural grid. The response was measured using EEG on the scalp. Subsequently, the brain-to-skull ratio was found by finding the best match between the computed and the measured scalp potentials, using the FEM forward solver.

Frequency and capacitive dependence. Several studies show that the conductivity of the skull depends on the frequency [70, 71, 72] and can affect the forward and inverse calculations considerably [194]. The frequency dependence of the conductivity of three layers of a live and dead skull was studied by Akhtari [8]. Their results indicate a frequency dependence that follows a power law for frequency in the range 10–90 H_z. However, the fitted parameters depend largely on the sample and whether the skull is dead or alive. The conductivity frequency dependence was also observed in the study by Tang [197], but the dependence was weaker for the EEG frequency range.

TABLE IV.1: Skull conductivity

Publication	Region	Conduct. S/m	Samples	Methods	Freq. H_z	T C^o
Law 1993	Suture Compact bone Trilayer bone	.0123-.074 .0047-.0078 .0100-.028	4 2 14	dead, soaked in .9% saline	100	
Oostendorp 2000	Uniform	0.015 0.013	1 2	Dead In vivo	1e5 10-1k	37 37
Akhtarai 2000	Top compact Spongy Bottom compact Bulk	0.0023 0.0077 0.0033 0.003	4	Dead, soaked in saline	10-90	
Akhtarai 2002	Top compact Spongy Bottom compact Bulk	0.0062 0.021 0.0049 0.0095	4	live	10-90	
Hoekema	Uniform	.032-.08	5	Live	10	37
2003						
Tang 2008	Std trilayer Qusai trilayer Std compact Qusai compact Dentate suture Squamous suture	0.013 0.0069 0.0038 0.005 0.017 0.0078	58 110 62 53 41 64	Live	1-4e6	36.5

Akhtari [9] studied the validity of the quasi-static approximation of ignoring the capacitive effect of the skull trilayer bones. Their results show that this approximation is adequate. These results confirm previous results obtain by Stinstra [194]. Therefore, the three-layers model of the skull can be modeled as a series of three resistors.

4.4 Finite Difference Discretization

As mentioned before, the MRI's segmented image can be used directly as the volume conductor model for the FDM method since the image voxels map directly to the regular cubic FDM grid. In contrast, the FEM method requires the creation of a tetrahedral mesh from the MRI image that defines the boundaries between different tissues. The accuracy of the FEM method is limited by the accuracy of constructing such a mesh, which is hard in complex geometry of the human head. Therefore, we chose the FDM-based method to solve Poisson's equation. The alternating direction implicit finite difference method (ADI) is an attractive FDM approach for solving elliptic and parabolic PDE. In ADI, solving the system requires a number of uncoupled tridiagonal solutions, which require little memory and can be executed concurrently on a multiprocessor machine. In this section, we descretize Poisson's equation in the multicomponent ADI scheme. In §4.4.1 we desceretize the right-hand side, the current source term, and in §4.4.2 we disceretize the left-hand side, the laplace operator.

4.4.1 Alternating Direction Implicit (ADI) Method

In numerical analysis, a common concept used for solving an elliptic problem is to add a first-time derivative to the PDE and then to solve the resulting parabolic equation until a steady state is reached. At the steady state, the time derivative is zero and the solution corresponds to the original problem. Based on this concept and assuming the source term, $\nabla \cdot \mathbf{J}_s = I_s$, Poisson's equation with the time derivative term is,

$$\frac{\partial \phi}{\partial t} + \nabla \cdot \sigma(x, y, z) \nabla \phi(x, y, z) = I_s. \tag{IV.16}$$

Then we used the iterative Multi-Component Alternating Directions Implicit (ADI) algorithm [4] to solve this equation. The method is a generalization of the classical Douglas-Rachford ADI algorithm [55], but with improved stability in 3D. It is unconditionally stable in 3D for any value of the time step. To describe the electrical conductivity within arbitrary geometry, we used the method of *embedded boundaries*. In this method, the object with arbitrary geometry is embedded into a cubic computational domain with zero conductivity in the complementary region. Therefore, no current is allowed to flow out of the physical area and the Neuman boundary condition is naturally satisfied. The idea of the iterative ADI method is that the solution of Equation IV.15 is the steady state solution of the corresponding time-dependent Poisson equation.

The finite-difference scheme is used over the solution domain by using a rectangular grid with spatial spacings of h_x, h_y, h_z in the x, y, z direction, respectively, measured in meters, and τ in the time direction, measured in seconds. Using the notation $x_i = ih_x$, $y_j = jh_y$, $z_k = kh_z$ and $t_n = n\tau$ for integer values of i, j, k and n, the electrical potential at a grid point, (i, j, k), at time, t_n, is written as, $\phi^n_{ijk} = \phi(x_i, y_j, z_k; t_n)$. The idea of the ADI method is that at every iteration time step the spatial operator is split into the sum of three 1D operators, which are evaluated alternatively at each sub-step. For example, the difference equation in x direction of Equation IV.16 is,

$$\frac{\phi_i^{n+1} - \frac{1}{3}(\phi_i^n + \phi_j^n + \phi_k^n)}{\tau} + \delta_x(\phi_i^{n+1}) + \delta_y(\phi_j^n) + \delta_z(\phi_k^n) = I_s, \tag{IV.17}$$

where τ is the time step and $\delta_x(\phi_i^{n+1})$, $\delta_y(\phi_j^{n+1})$ and $\delta_z(\phi_k^{n+1})$ are the descrete form of the partial derivative,

$$\delta_x(\phi_i^{n+1}) = \frac{1}{{h_x}^2}[\sigma_{i+1/2}(\phi_{i+1}^{n+1} - \phi_i^{n+1}) - \sigma_{i-1/2}(\phi_i^{n+1} - \phi_{i-1}^{n+1})],$$

$$\delta_y(\phi_j^{n+1}) = \frac{1}{{h_y}^2}[\sigma_{j+1/2}(\phi_{j+1}^{n+1} - \phi_j^{n+1}) - \sigma_{j-1/2}(\phi_j^{n+1} - \phi_{j-1}^{n+1})],$$

$$\delta_z(\phi_k^{n+1}) = \frac{1}{{h_z}^2}[\sigma_{k+1/2}(\phi_{k+1}^{n+1} - \phi_k^{n+1}) - \sigma_{k-1/2}(\phi_k^{n+1} - \phi_{k-1}^{n+1})].$$

Such a scheme is accurate to $O(\tau^2) + O(h_x^2 + h_y^2 + h_z^2)$. In contrast with the classic ADI method, the multi-component ADI uses the regularization (averaging) for evaluation of the variable at the previous instant of time. Rearranging Equation IV.17 we obtain,

$$\phi_i^{n+1} + \tau\delta_x(\phi_i^{n+1}) = \frac{1}{3}(\phi_i^n + \phi_j^n + \phi_k^n) + \tau[I_s - \delta_y(\phi_j^n) - \delta_z(\phi_k^n)]. \tag{IV.18}$$

Substituting $\delta_x(\phi_i^{n+1})$ into Equation (IV.18),

$$\begin{aligned}\phi_i^{n+1} + \frac{\tau}{h_x}[\sigma_{i+1/2}(\phi_{i+1}^{n+1} - \phi_i^{n+1}) - \sigma_{i-1/2}(\phi_i^{n+1} - \phi_{i-1}^{n+1})] &= \frac{1}{3}(\phi_i^n + \phi_j^n + \phi_k^n) \\ &\quad +\tau[I_s - \delta_y(\phi_j^n) - \delta_z(\phi_k^n)],\end{aligned}$$

where $\sigma_{i+1/2,j,k} = (\sigma_{i,j,k} + \sigma_{i+1,j,k})/2$ is the conductivity at $ih_x + 1/2$. The conductivity at the points between the grid points. Similar expresions for the y and z components can be obtained. Rearanging Equation IV.19,

$$1 - \frac{\tau}{h_x}(\sigma_{i+1/2} + \sigma_{i-1/2})]\phi_i^{n+1} + \frac{\tau}{h_x}\sigma_{i+1/2}\phi_{i+1}^{n+1} + \frac{\tau}{h_x}\sigma_{i-1/2}\phi_{i-1}^{n+1} = \phi_{ijk}^n + \tau[I_s - \delta_y(\phi_j^n) - \delta_z(\phi_k^n). \tag{IV.19}$$

The left hand side of Equation IV.19 is a tridiagonal system in the $(n+1)$ time step, while the right hand side is evaluated in the n time step. Let,

$$d_i = 1 - \frac{\tau}{h_x}(\sigma_{i+1/2} + \sigma_{i-1/2}),$$

$$a_i = \frac{\tau}{h_x}\sigma_{i+1/2},$$

$$c_i = \frac{\tau}{h_x}\sigma_{i-1/2},$$

$$b_i = \phi_{ijk}^n + \tau[I_s - \delta_y(\phi_j^n) - \delta_z(\phi_k^n)].$$

Then Equation IV.19 can be written,

$$a_i\phi_{i+1}^{n+1} + d_i\phi_i^{n+1} + c_i\phi_{i-1}^{n+1} = b_i^n. \tag{IV.20}$$

Assuming the dimensions of the computational grid are N_x, N_y, N_z, then the N_yN_z tridiagonal systems can be solved concurrently. Similar expressions for the y and z components can be obtained,

$$a_j\phi_{j+1}^{n+1} + d_j\phi_j^{n+1} + c_j\phi_{j-1}^{n+1} = b_j^n. \tag{IV.21}$$

$$a_k\phi_{k+1}^{n+1} + d_k\phi_k^{n+1} + c_k\phi_{k-1}^{n+1} = b_k^n. \tag{IV.22}$$

The above three equations are to be solved in the above order. For example, in evaluating b_j^n we use the updated value of ϕ_i^n which is actually ϕ_i^{n+1}, not the value of ϕ_i^n in the previous time step. I_s is the current source discretized in the §4.4.2. These are tridiagonal equations which can be solved iterativily using Thomas algorithm (tridiagonal algorithm).

4.4.2 Current Source Model

The right side of Poisson's equation, $\nabla \cdot \mathbf{J}_s$, represents the current density of the sources. Applying the divergence operator to the vector field $\mathbf{J}_s$ [86],

$$\nabla \cdot \mathbf{J}_s = \lim_{V\to 0} \frac{1}{\mathbf{V}} \oint_{\partial V} \mathbf{J}.\mathbf{n}\; ds = I_s,$$

we obtain the current density I_s in (A/m^3). The surface integral represents the net current (in A) leaving or entering the volume V. By definition, the current direction is that of positive charge movement. Therefore, the integral is positive when a net current leaves the volume (current source), negative when a net current enters the volume (current sink), and zero when the leaving and entering currents are equal [140]. In another way, a current source adds positive charges to the extracellular volume and a current sink removes positive charges from the extracellular volume. Since there is no pile up of charges inside the brain, the total current sources and the total current sinks are equal.

As discussed before, the currents behind EEG/MEG signals are those due to postsynaptic potentials. In case of EPSP, the current sink corresponds to removal of positively charged ions from the extracellular environment (sodium ions) at the apical dendrite, while the current source corresponds to injection of positively charged ions at the cell body. If we consider that a volume element V encloses a current sink at position $\mathbf{r}_1$, then the surface integral of the current density is $-I$. As the volume V goes to zero, the total current remains $-I$, but the division by the volume gives a singularity. Therefore, the current density at $\mathbf{r}_1$ can be written in terms of the dirac delta function as $-I\delta(\mathbf{r}-\mathbf{r}_1)$, where $\delta(\mathbf{r}-\mathbf{r}_1)$ replaces the limit $\lim_{V\to 0}\frac{1}{\mathbf{V}}$. By similar reasoning the current source at position $\mathbf{r}_2$ at the soma can be written as $I\delta(\mathbf{r}-\mathbf{r}_2)$. In the case of IPSP the current source is at the apical dendrite while the current sink is at the soma. Current sources and sinks are monopole because they correspond to one end of the current, assuming the other end is unknown. Hence, in terms of monopoles, the right-hand side of Poisson's

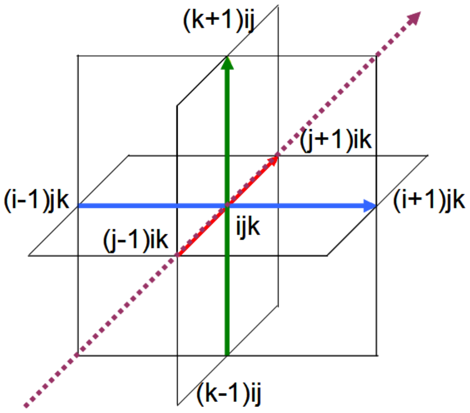


FIGURE 4.1: A current dipole moment, $\mathbf{q} = Id\mathbf{e}$, can be expressed in terms of three orthogonal dipole moments.

equation becomes,

$$\nabla \cdot \mathbf{J}_s = I(\delta(\mathbf{r} - \mathbf{r}_2) - \delta(\mathbf{r} - \mathbf{r}_1)).$$

In this expression, the delta function replaces the limit, $\lim_{V\to 0} \frac{1}{\mathbf{V}}$, where V is the volume around the monopole. When taking the limit in the FDM descritization the volume goes to the volume of the voxel that contains the monopole, $V = h_x \cdot h_y \cdot h_z$. Therefore, the delta function in 3D is,

$$\delta(\mathbf{r} - \mathbf{r}') = \begin{cases} \frac{1}{h_x \cdot h_y \cdot h_z} & \text{if } \mathbf{r} = \mathbf{r}' \\ 0 & \text{if } \mathbf{r} \neq \mathbf{r}' \end{cases},$$

where h_x, h_y, and h_z are the mesh spacing. In the bEIT inverse problem, the locations of the current sink $\mathbf{r}_1$ and current source $\mathbf{r}_2$, and the magnitude of the injection current I are given. Therefore, the right-hand side of Poisson's equation (Equation IV.15) is modeled as two current monopoles.

In EEG source modeling, the distance between the two monopoles is small compared to the observation distance and the two monopoles look like a current dipole source. Therefore, the current source and the current sink are typically represented by a current dipole with a position chosen at halfway between them. Similar to an electrostatic dipole, the current dipole moment $\mathbf{q}$ is defined by a vector $q\mathbf{e}$, which is directed from the current sink to the current source with magnitude, $q = ||\mathbf{q}|| = Id$, where d is the distance between the two monopoles, therefore, $\mathbf{q} = Id\mathbf{e}$. In EEG source analysis, the dipole moment is given and the current density is $I = q/d$. Then, in terms of the dipole moment, the right-hand side of Poisson's equation is,

$$\nabla \cdot \mathbf{J}_s = \begin{cases} \frac{q}{dh_xh_yh_z} & \text{if } \mathbf{r} = \mathbf{r}_{source} \\ -\frac{q}{dh_xh_yh_z} & \text{if } \mathbf{r} = \mathbf{r}_{sink} \\ 0 & \text{otherwise} \end{cases},$$

In EEG source analysis, it is convenient to express the dipole moment $\mathbf{q}$ in terms of its cartesian components, $\mathbf{q} = q_x\mathbf{e}_x + q_y\mathbf{e}_y + q_z\mathbf{e}_z$, which allow encoding the dipole direction as well as its magnitude in the FDM model. In this formulation, given a dipole moment, $\mathbf{q}_{ijk}$, at position ijk, it can be expressed in terms of three orthogonal dipole moments along x-, y-, and z-axises in terms of monopole sources at $(i+1)jk$, $i(j+1)k$, $ij(k+1)$ and monpole sinks at $(i-1)jk$, $i(j-1)k$, $ij(k-1)$, respectively. Then the current source term can be expressed in terms of these

TABLE IV.2: Tissues parameters in 4-shell models[61].

Tissue type	$\sigma(\Omega^{-1}m^{-1})$	Radius(cm)	Reference
Brain	0.25	8	Geddes(1967)
Csf	1.79	8.2	Daumann(1997)
Skull	0.018	8.7	Law(1993)
Scalp	0.44	9.2	Burger(1943)

dipoles as shown in Figure 4.1,

$$\begin{aligned}\nabla \cdot \mathbf{J}_s &= \frac{q_x}{2h_x^2 h_y h_z}(\delta_{(i+1)i'} - \delta_{(i-1)i'})\delta_{jj',kk'} + \frac{q_y}{2h_x h_y^2 h_z}(\delta_{(j+1)j'} - \delta_{(j-1)j'})\delta_{ii',kk'} \\ &+ \frac{q_z}{2h_x h_y h_z^2}(\delta_{(k+1)k'} - \delta_{(k-1)k'})\delta_{ii',jj'},\end{aligned}$$

where $\delta_{xx'}$ is the Kronecker delta function. Similar expressions for the y-, and z-axis dipoles can be writen.

4.5 Evaluations and Results

We have built a finite difference forward problem solver for Equation IV.15 based on the multicomponent alternating directions implicit (ADI) Algorithm §4.4.1. In the following subsections, we discuss the verifications of the forward solver in §4.5.1. Then we discuss some applications and use of the forward solver in §4.5.2. In §4.5.3 we discuss parallelizing the forward solver and porting it onto the cell broadband engine [104].

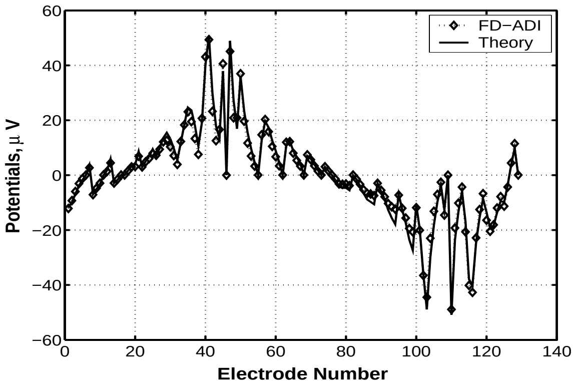


FIGURE 4.2: Verification of the forward solver accuracy against analytics for a 4-shell spherical model.

4.5.1 Forward Solver Verification

The forward solver was tested and verified against a 4-shell spherical model, as well as low ($4mm$) and high ($1mm$) resolution human MRI data. For comparison purposes, we considered that spherical geometry consists of four tissue types. Their values were set to those in the spherical model (Table IV.2). Then we computed potentials at standard locations for the 129 electrodes configuration montage on the spherical model and compared the results with the analytical solution [61] available for a 4-shell spherical model in Figure 4.2. One can observe good agreement, save for some minor discrepancies (average error is no more than a few percents) caused by the mesh orientation effects (the cubic versa spherical symmetry).

4.5.2 Forward Solver Application

Having a forward solver that supports realistic geometries allows us to perform a series of computations for the electrical potentials and currents inside a

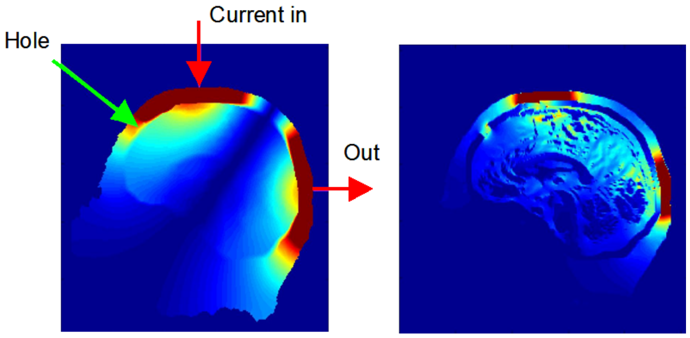


FIGURE 4.3: Simulating a hole in the skull. The potential distribution (left) and the current (right).

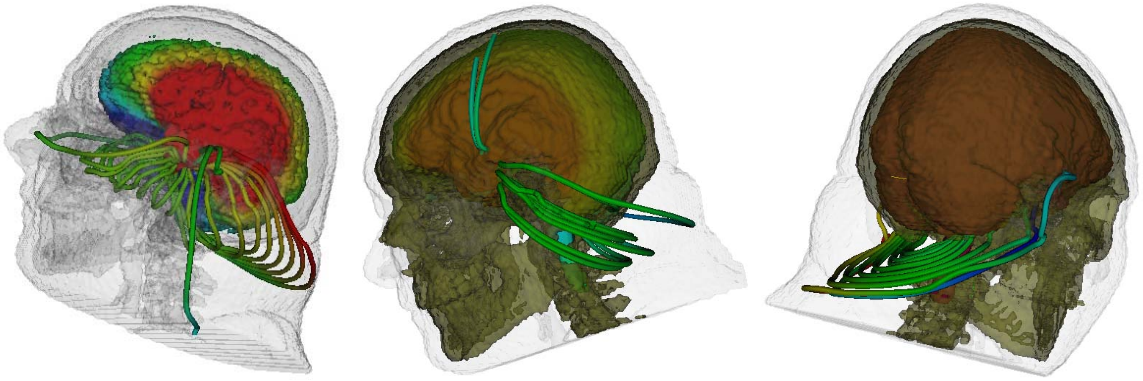


FIGURE 4.4: Current and potential distributions due to a current dipole source in the cortex; holes in the skull allow more current to flow through the skull and reach the scalp, which increases the effective conductivity of the skull.

human head with a surgical or traumatic opening in the skull (Figure 4.3). We found that generally low resolution ($64 \times 64 \times 44$ voxels) is not enough for accurate description of the current and potentials distribution through the head, as the coarse discretization creates artificial shunts for currents (mainly in the skull). With increased resolution ($128 \times 128 \times 88$ or $256 \times 256 \times 176$ voxels) our model has been shown to be capable of capturing the fine details of current/potential redistribution caused by the structural perturbation. However, the computational requirements of the forward calculation increase significantly.

Also, we used the forward solver to simulate the EEG data corresponding to a dipole current source placed on the cortical surface. Figure 4.4 shows the current and potentials distributions through out the volume conductor. As the figure shows, holes in the skull allow more current to penetrate the skull and reach the scalp, which increases the effective conductivity of the skull. This indicates how important it is for a head model to capture the holes in the skull, contrary to spherical and BEM models, where holes are inherently ignored.

It is also worth noting, that the ADI algorithm can be easily adapted for solving PDEs describing other tomographic modalities. In particular, we have used it in other related studies–for example , in simulation of photon migration (diffusion) in a human head in near-infrared spectroscopy of brain injuries and hematomas.

4.5.3 Parallelizing the Forward Solver

A typical head modeling scientific investigation–e.g., solving the source localization inverse problem, solving the conductivity inverse problem and performing sensitivity analysis–involves intensive execution of the forward problem. Therefore, an efficient portable forward solver is necessary to enable such investigations. Improving the performance of head-modeling investigations can be achieved by improving the performance of the forward solver and by finding concurrency in computing the forward solution. In this section, we describe the forward solver parallelism in shared memory architecture and also we ported it onto the cell broadband engine for potential access to cell blade clusters. The goal here is to make the forward solver as efficient as possible and as portable to different computing resources as possible. In Chapter VI, we provide the design of a generic framework that can use any available computing resources to perform scientific investigations.

The ADI algorithm consists of a time iteration loop in which each time step is split into three substeps §4.4.1. In each substep a tridiagonal system of equations is solved along either x,y or z direction. For instance, in the first substep the spatial operator acts only on the x direction. So all $N_y N_z$ equations along the x-direction are independent and can be solved concurrently. Similarly, in the second substep, all $N_x N_y$ equations along the y-direction and, in the third substep, all $N_x N_y$ along the z-directions are independent and can be solved concurrently. However, at the end of each substep all equations must be solved before proceeding to the next substep. Therefore, at the end of each substep all processors must be synchronized before proceeding to the next substep.

Shared Memory Architecture

Parallelization of the ADI algorithm is straightforward in shared memory architecture, where the time loop runs sequentially and then in each substep all

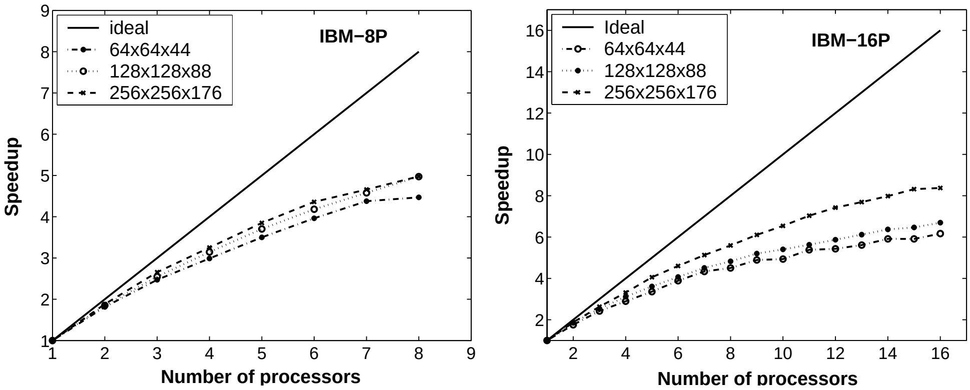


FIGURE 4.5: Speed-up of the forward solver for different problem sizes on 8-processor (left) and a 16-processor (right) IBM machines.

processors cooperate in solving the independent tridiagonal systems of equations concurrently. The parallel algorithm pseudo code is shown below:

```
While (not termination condition)
      Solve NyNz systems of equations
      Barrier
      Solve NxNz systems of equations
      Barrier
      Solve NxNy systems of equations
      Barrier
```

The forward solver was parallelized using OpenMP. The performance speedups for 64 × 64 × 44, 128 × 128 × 88 and 256 × 256 × 176 sized problems on the IBM p655 (8 processors) and p690 (16 processors) machines are shown in Figure 4.5. The importance of understanding the speedup performance on the

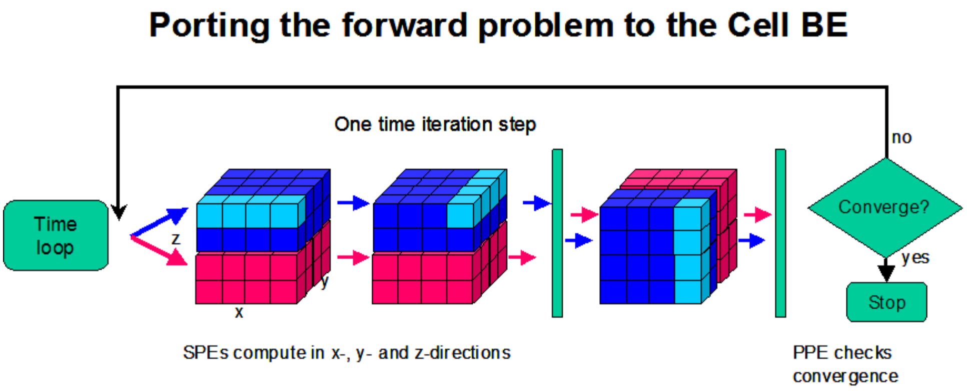

Porting the forward problem to the Cell BE
One time iteration step
Time loop
z
y
x
no
Converge?
yes
Stop
SPEs compute in x-, y- and z-directions
PPE checks convergence

FIGURE 4.6: PPE controls the time loop and SPEs do the actual computations.

cluster compute servers is to allow flexible allocation of resources between inverse and forward processing.

Cell Broadband Engine

The Cell Broadband engine[104] is an example of a heterogeneous multicore processor. It includes one PowerPC processor (PPE) and eight synergistic processors (SPEs). The PPE serves as a general purpose multithreaded processor and as a scheduler for the computation on the SPEs.

We have ported the FDM ADI algorithm to the Cell Broadband Engine. We used an IBM SDK2.0 programming environment. Our implementation was tested and validated against a 4-shell spherical model on Play Station 3. Our results show a decent performance compared to a shared memory implementation on the p655 reference cluster. Porting the ADI algorithm to the Cell BE is achieved as shown in Figure 4.6:

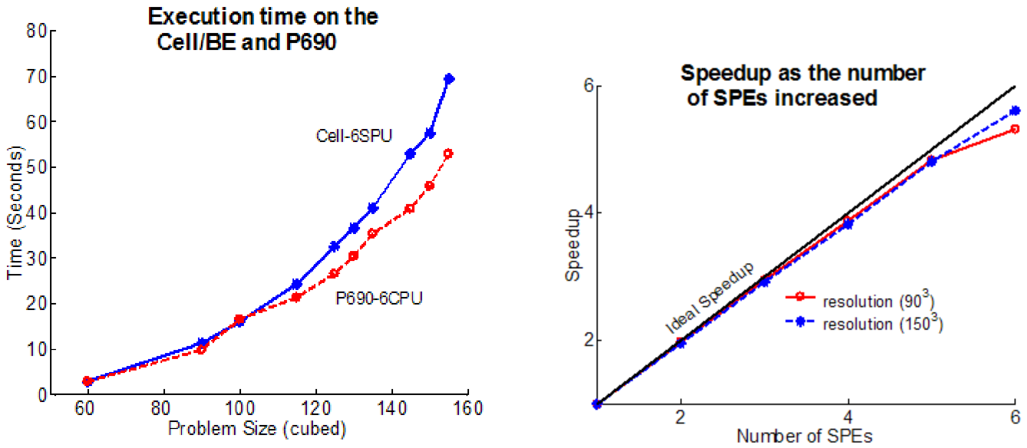


FIGURE 4.7: Performance of the forward solver on the Cell Be. Comparison with a shared memory implementation (left) and Speedup curve (right).

1. The PPE processor executes the time iteration loop and checks the convergence condition at the end of every time step.

2. Each time iteration step is split into three substeps. In the first substep all SPEs cooperate in solving the independent N_yN_z x-direction tridiagonal systems (Equation IV.21).

3. Similarly, in the second and third substeps, all SPEs solve the N_xN_z and N_xN_y tridiagonal systems in y and z directions.

4. At the end of the second and third substeps, all SPEs are synchronized. The synchronization after the first substep is not required, as the computational grid is decomposed along the z-direction.

5. The computation proceeds on one bar of the computational grid at a time.

6. Overlapping computation with DMA transfer is achieved using three buffers. While the computation proceeds from one buffer, a DMA transfer from the already computed buffer is posted. At the same time a DMA transfer is posted to bring data that are needed for the next computation to the third buffer.

The Cell forward solver implementation was tested on PS3 for several problem sizes. As shown in Figure 4.7, the performance on PS3 is competing with an openMP implementation on the IBM p655 using the same number of CPUs, especially for larger problem sizes. However, due to the memory limitations of 250 MB on the PS3, we can process head of dimensions only up to (120^3).

4.6 Summary and Conclusions

We have built an efficient and robust 3D Poisson solver based on a finite difference ADI algorithm for modeling electrical problems in heterogeneous biological tissues. We focus in particular on modeling the conductivity properties of the human head.

Developing an accurate head model has been and still is under active research. Recent advances in the geometrical head models and the use of realistic geometry have been very important. However, they are insufficient as long as the conductivity model is not accurate. In the meantime, the most important factor in head modeling is to obtain an accurate conductivity model of the skull. Published data about skull conductivity values cover a wide range and are highly controversial, as summarized in Table IV.1. These large variations in skull conductivity values expose the complexity of the skull structure.

The reasons behind these variations can be related to two main factors. First, these data are obtained for different subjects with different ages, and it is expected that the conductivity of the skull will vary across subjects and will vary with age within the same subject. This is expected because the skull gets dryer with age and skull variations change with age and across subjects. The second factor is related to the procedure of measuring or estimating skull conductivities.

The main two procedures used in estimating or measuring the conductivity of the skull is in-vitro and in-vivo. In in-vitro measurements, the conductivity of samples of dead skull soaked in saline or samples from live skulls are measured. The measurement procedure and condition–e.g., temperature and humidity–are expected to affect the results. Also, the conductivity of live skull is expected to be different from the conductivity of dead hydrated skull because live skull contains natural body fluid while dead hydrated skull contains saline. In the in-vivo conductivity estimation using a scalp current injection, it was observed that the conductivity is highly model-dependent. Different results are obtained when different head models are used for the same data. Even more significant, in simulation it was shown that scalp potential distributions can be obtained using 3-shell and 4-shell models with different brain-to-skull conductivity ratios [140].

Therefore, it is not surprising to observe significant variations in published data about skull conductivity values. To ascertain the accurate volume conductor of a subject, one must determine the subject's own geometry and conductivity values. Obviously, a noninvasive approach must be used. In the meantime, EIT impedance imaging provides the best approach to use.

In all previous in-vivo studies, some kind of oversimplification of the problem was used. In all the studies, the inverse impedance-mapping problem was oversimplified to fit only one parameter. In most studies, spherical models are used and in the rest a BEM is used. Spherical models inherently ignore the geometrical variation of the skull. Both spherical and BEM models treat the skull as a single uniform tissue and ignore all the important structural variations of the skull, including holes.

In this dissertation, we provide the necessary computational tools and a volume conductor model that supports capturing the main skull variations, including skull holes and structural variations. We leveraged HPC to overcome the computational performance instead of oversimplifying the problem. The next chapter formulates the inverse impedance mapping problem and the inclusion of skull variations by parcellation.

CHAPTER V

ELECTRICAL IMPEDANCE TOMOGRAPHY

Electrical impedance tomography (EIT) [26] [98] is a widely used non-invasive imaging method. In this method, the conductivity or the permitivity of a body tissue is determined from electrical measurements on the surface. Data acquisition is performed using array of conducting electrodes attached to the surface of the subject's skin surrounding the body part under the study and a small alternating current are applied to some of the electrodes. The resulting electrical potential on the rest of the electrodes are measured. The process is repeated for several different configurations (Figure 5.1). Then it is possible to estimate the internal conductivity distribution of the body part from the boundary data by solving the EIT *inverse problem.*

EIT as an imaging method is based on the fact that the conductivity or permitivity of biological tissue varies between tissue types and depends on other factors such as temperature and physiological factors. The method is sensitive to changes in tissue electrical conductivity. Therefore, it is possible to construct a map

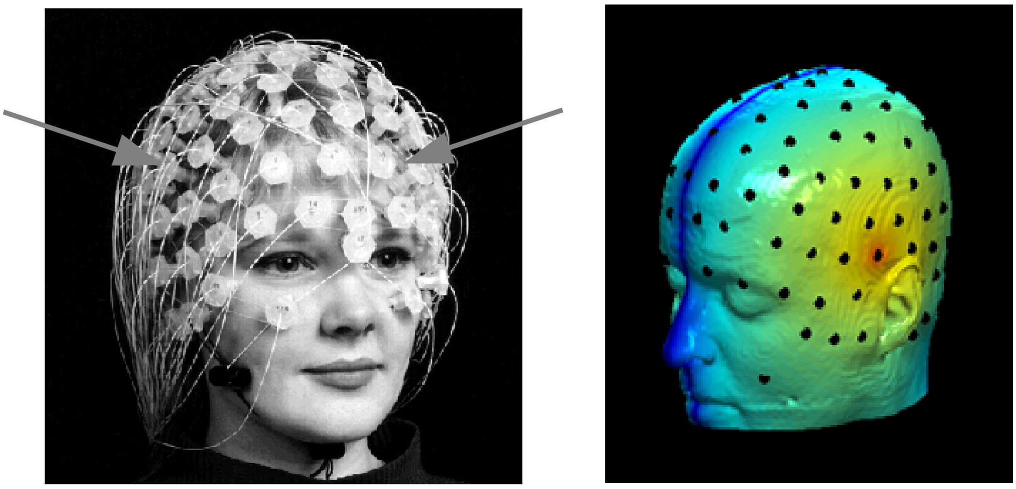


FIGURE 5.1: EIT measurements, small alternating current injected into pair of electrodes and the response is measured at the rest of the electrodes. Data acquisition set up (left) and the computational model (right).

of the conductivity of the region of the body propped by the current. The method relatively has poor resolution compared to other imaging methods such MRI or CT. The resolution is highly controlled by the number of electrodes. But EIT is the only method that measures conductivity directly and it is inexpensive and fast. EIT has several application includes detection of skin and breast cancer [192], localization of epileptic foci [13] [100] and monitoring lungs function [66].

In head modeling the geometry of the head is typically constraints to a few tissue types imposed by the segmented MRI data. The aim of estimating the head tissue conductivities in the head modeling is to create an accurate forward model for the EEG source localization while in EIT the aim is to reconstruct a conductivity image to differentiate between tissue types. Therefore, in head modeling one needs to know the average regional conductivities of a few tissue types, for example, scalp,

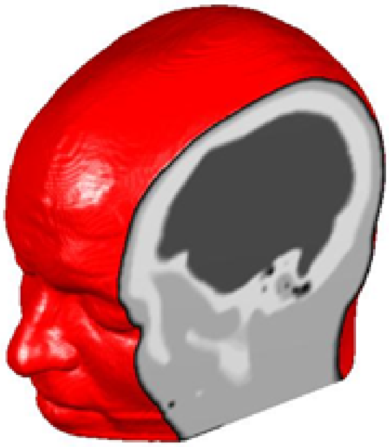


FIGURE 5.2: In bEIT one needs to know the average regional conductivities of a few tissue types (Scalp, Skull, Brain).

skull, cerebrospinal fluid (CSF) and brain (Figure 5.2) contrary to EIT where one needs to reconstruct the back end conductivity image. This significantly reduces the dimensionality of the parameter space in the inverse search as well as the number of iterations in converging to a global minimum. We call this method *bounded Electrical Impedance Tomography (bEIT)* and we use this approach to estimate the conductivities of the head tissues using FDM forward solver.

5.1 Conductivity Estimation

In EIT and bEIT the current paths through the tissues are not well defined. The current propagates though diffusive ionic interaction in the body tissues. Therefore, recovering the body conductivity distribution from the boundary potentials is a non-linear inverse problem. Further, the bEIT and EIT inverse problem are ill-posed in the sense that small errors in the measured data can cause large errors in the estimated conductivities and the solution becomes unstable.

EIT is based on the principle of the back end image reconstruction. The goal is to reconstruct the internal conductivity or permitivity distribution efficiently and as accurately as possible in two or three-dimensional models. The EIT image reconstruction problem is mathematically formulated as an optimization problem where a suitable objective function $F(\sigma)$ is minimized. The objective function is typically composed from two parts. One part corresponds to the goodness of the fit between the computed potentials using the forward model and the measured data. The second part is to overcome the ill-posedness of the problem which includes additional a *priori* information to regularize and stabilize the solution [113][120]. Typically, the standard Tikhonov regularization methods as described in [25] are used. Then a deterministic approach based on the Least Square Method is usually used to minimize an objective function of the form $F(\sigma)$,

$$F(\sigma) = \sum \|U_{measured} - U_{computed}(\sigma)\|^2 + \alpha \|L\sigma\|, \tag{V.1}$$

where σ is the unknown internal conductivity vector, $U_{measured}(\sigma)$ is the vector of measured potentials on the boundary of the object, $U_{computed}(\sigma)$ is the vector of computed potentials with respect to σ using the forward model (normally FEM), α

is a regularization parameter and L is a regularization matrix connecting adjacent elements of different conductivities.

To solve Equation V.1, an iterative minimization method such as Newton-Raphson method is commonly used for its fast convergence and good reconstruction quality. However, the method is a local minimization method and it is likely to be trapped in local minimum and normally additional regularizations are used to restore stability. The stability of the method is highly dependent on the choice of the parameter α and the initial values of the conductivities.

The general formulation of the inverse problem in bEIT is the same as in EIT where the problem is formulated as a minimization problem of a suitable objective function that measures the goodness of the fit between the computed potentials and the measured potentials. However, since the number of parameters in bEIT inverse problem is few, the procedure for bEIT conductivity extraction is different. In bEIT a stochastic global minimization algorithm such as Simulated Annealing can be used to minimize the objective function. To overcome the ill-posdness of the problem, additional a *priory* information can be used in a form of constraints. The mathematical formulation of the bEIT problem can be described as follows. From the assumed distribution of the head tissue conductivities σ_{ij}, and the given injection current configuration, S, it is possible to predict the set of potentials measurement values, ϕ, given forward model F, as the nonlinear functional,

$$\phi^p = F(\sigma(x, y, z)). \tag{V.2}$$

Then an appropriate objective function is defined, which measures the error between the measured, V and predicted ϕ^p, and a search for the global minimum is undertaken using advanced nonlinear optimization algorithm under imposed constraints. In the following sections we discuss the candidate objective functions that can be tested and the non-linear optimization algorithms we used in our research.

5.2 Objective Functions

The objective function quantifies the difference between the numerically computed potential values and the measured data. There are several metrics that can be used to measure the difference between two sets of data. These metrics can be used as objective functions. In this section we describe these metrics as reference in the rest of the dissertation. All our simulated analysis is based on the L2 norm metric. However, it is beneficial to study other metrics when working with real data.

Root Mean Square Error (L2 norm) L2 norm is the most popular metric that is used to quantify the difference between two sets of data. This measure is computed by taking the average of the square differences between each computed value and its corresponding measured value. The root mean-square error is simply the square root of the mean-squared error and it takes the mathematical form,

$$E = \left(\frac{1}{N} \sum_{i=1}^{N} (\phi_i^p - V_i)^2 \right)^{1/2}, \tag{V.3}$$

where N is the number of measuring sensors (electrodes), ϕ_i and V_i are computed and measured potentials, respectively, at sensor i.

Relative Root Mean Square Error (rRMS) rRMS is based on the relative difference between the computed and the measured data instead of the absolute values. The rRMS fitness function is based on the standard relative mean square error,

$$relE = \sqrt{\frac{\sum_{i=1}^{N} (\phi_i - V_i)^2}{\sum_{i=1}^{N} \phi_i^2}}. \tag{V.4}$$

Magnitude Factor (MAG) and the Relative Difference Measure (RDM) The *Magnitude Factor* MAG and the *Relative Difference Measure* RDM metrics are defined in [126] and commonly used to compare between two sets of data,

$$\begin{aligned} MAG &= \sqrt{\frac{\sum_{i=1}^{N} (V_i)^2}{\sum_{i=1}^{N} (\phi_i)^2}}, \\ RDM &= \sqrt{\sum_{i=1}^{N} \left[\frac{V_i}{\sqrt{\sum_{i=1}^{N} (V_i)^2}} - \frac{\phi_i}{\sqrt{\sum_{i=1}^{N} (\phi_i)^2}} \right]}, \end{aligned}$$

where ϕ_i and V_i denote two sets of potentials at the electrodes. The MAG measures the gain between two sets of data. The RDM measures the pattern variation (shape) independent of the magnitude. Two equal data sets gives $MAG = 1$ and $RDM = 0$.

5.3 Global Optimization Algorithms

Global optimization is the problem to find the globally best optimal solution of a model in the presence of multiple local optima. The optimization problem can be stated as follows: Find the values of the best optimal parameters that optimizes (minimizes or maximizes) the objective function and subject to some constraints. The maximization of a problem can be treated as the minimization of its negative. The optimization problem takes the following form,

$$Optimize(F(x)) \quad \text{subject to } l < x < u,$$

where l, u are the lower and upper bounds and F is the objective function to be optimized. Finding a local minimum to a problem is considered to be straightforward using some local optimizer. However, finding the global minimum is a challenge. A significant amount of research has been done in the past few decades and several algorithms are developed. Some of the popular algorithms are Genetic Evolutionary Algorithms, Tabu Search and Simulated Annealing.

In the study of this thesis, early in our work we chose Simplex Search algorithm using multi-start technique for its rapid prototyping to test the feasibility of our approach. Simplex search performed well in extracting up to four tissue conductivities §5.4.2, However, it failed to extract more than six tissue conductivities (required to improve the accuracy of the forward solver) in a reasonable amount of time. Therefore, we opted Simulated Annealing algorithm for its proven robustness in several applications. Choosing the best optimization algorithm for a particular problem is problem dependent, and the best way to choose an algorithm is to try them all[213]. This is one motivation behind developing ODESSI framework, discussed in Chapter VI, to allow a scientist to experiment with different algorithms and objective functions in ease. In sections §5.4 and §5.5 we discuss the algorithms, the computational environments and the results based on the simplex search and simulated annealing optimization.

5.4 Conductivity Modeling - Simplex Search

To solve the nonlinear optimization problem in Eq. V.1, early in our work we employed the downhill simplex method of Nelder and Mead as implemented by Press et al[156]. To avoid the local minima, we used a statistical approach. The inverse procedure was repeated for hundreds sets of conductivity guesses from appropriate fisiological intervals, and then the solutions closest to the global minimum solution were selected using the simple critirea $E < E_{threshold}$. We

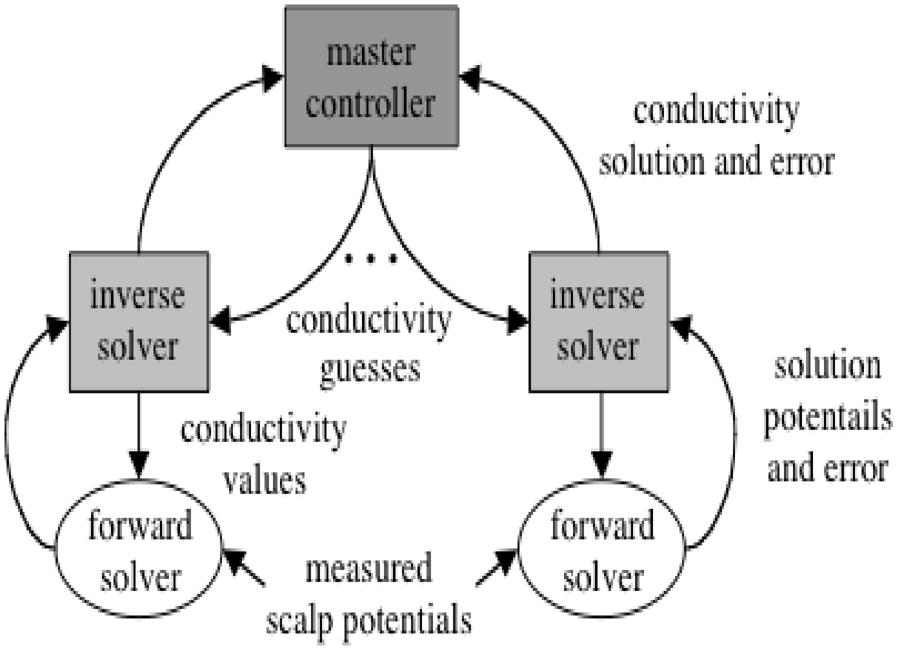


FIGURE 5.3: Schematic view of the parallel computational system based on simplex search optimization.

parallelized the simplex search using a multi-start technique as described in section §5.4 and good speedup was achieved.

5.4.1 Computational Design

The solution approach maps to a hierarchical computational design that can benefit both from parallel parametric search and parallel forward calculations. Figure 5.3 gives a schematic view of the approach we applied in a distributed environment of parallel computing clusters. The master controller is responsible for launching new inverse problems with guesses of conductivity values. Upon completion, the inverse solvers return conductivity solutions and error results to the master. Each inverse solver runs on a compute server. Given N compute servers, N inverse solves can be simultaneously active, each generating forward problems that can run in parallel, depending on the number of processors available. The system design allows the number of compute servers and the number of processors per server to be decided prior to execution, thus trading off inverse search parallelism versus forward problem speedup.

At our group (at the time of conducting these studies), we had access to a computational systems environment consisting of four multiprocessor clusters. Clusters *Clust1*, *Clust2*, and *Clust3* are 8-processor IBM p655 machines and cluster *Clust4* is a 16-processor IBM p690 machine. All machines are shared-memory multiprocessors running the Linux operating system. The clusters are connected by a high-speed gigabit Ethernet network. In our experiments below, we treated each machine as a separate compute server running one inverse solver. The forward problem was parallelized using OpenMP and run on eight (*Clust1-3*) and sixteen (*Clust4*) processors. The master controller can run on any networked machine in the environment. In our study, the master controller ran on Clust2.

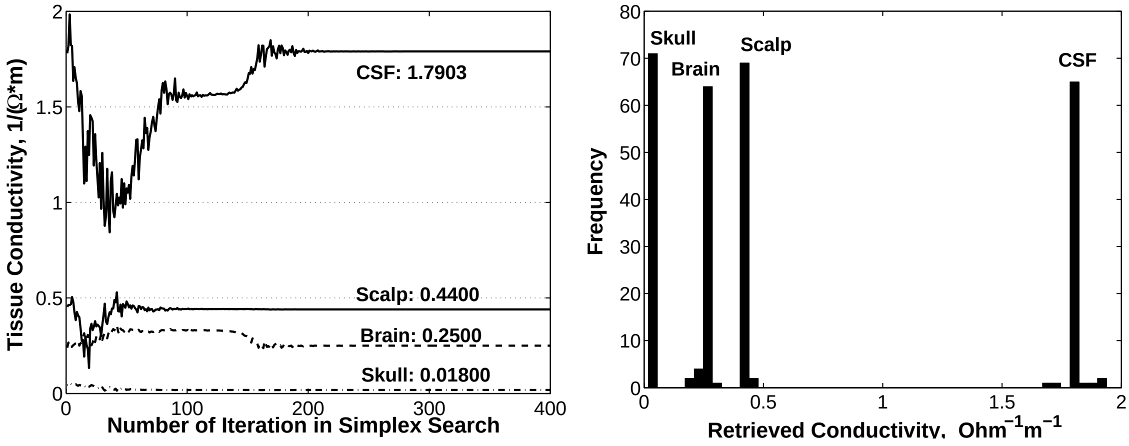


FIGURE 5.4: Results of the inverse search. Dynamics of the individual search (left) and statistics of the retrieved conductivities for about 200 initial random guesses(right). The actual number of the solutions shown is 71, their error function is less than 1 μV.

5.4.2 Results

First we evaluated the feasibility and performance of the inverse solver using simulated data. Then we performed a numerical study to extract three tissues with realistic data.

Simulated Data

In the inverse search the initial simplex was constructed randomly based upon the mean conductivity values (Table IV.2) and their standard deviations as it is reported in the related biomedical literature. In the present test study we did not use the real experimental human data, instead, we simulated the experimental set of the reference potentials V in Equation V.1 using our forward solver with the mean conductivity values from Table IV.2, which had been assumed to be true, but not

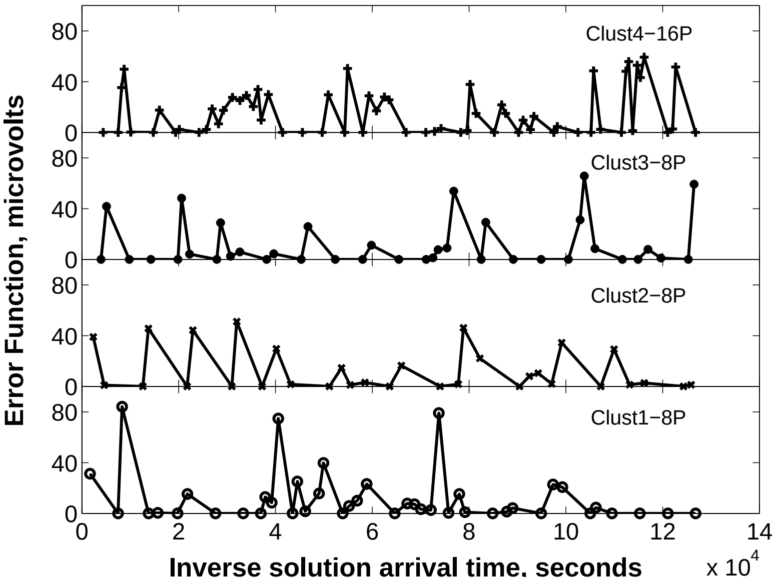


FIGURE 5.5: Solution flow at the master controller. Inverse solution arrival to the controller are marked.

known a priory for a user running the inverse procedure. The search was stopped when one or two criteria were met. The first is when the decrease in the error function is fractionally smaller than some tolerance parameter. The second is when the number of steps of the simplex exceeds some maximum value. During the search, the conductivities were constrained to stay within their pre-defined plausible ranges. If the simplex algorithm attempted to step outside of the acceptable range, then the offending conductivity was reset to the nearest allowed value. Our procedure had the desired effect of guiding the search based on prior knowledge. Some number of solution sets included conductivities that were separated from the bulk of the distribution. These were rejected as outliers, based on the significant larger square error norm in Equation V.1 (e.g., the solution sets were filtered according to the criteria $E < E_{threshold}$). We have found empirically that setting $E_{threshold} = 1\mu V$ in most of our runs produced a fair percentage of solutions close to the global minimum.

The distribution of the retrieved conductivities is shown in Figure 5.4 (right). The fact that the retrieved conductivities for the intracranial tissues (CSF and brain) have wider distributions is consistent with the intuitive physical explanation that the skull, as having the lowest conductivity, shields the currents injected by the scalp electrodes from the deep penetration into the head. Thus, the deep intracranial tissues are interrogated less in comparison with the skull and scalp. The dynamics of an individual inverse search convergence for a random initial guesses is shown in Figure 5.4 (left). After filtering data according to the error norm magnitude, we fitted the individual conductivities to the normal distribution. The mean retrieved conductivities $\sigma(\Omega^{-1}m^{-1})$ and their standard deviations $\Delta\sigma(\Omega^{-1}m^{-1})$ are: Brain (0.24 / .01), CSF (1.79 / .03), Skull (0.0180 / .0002), and Scalp (0.4400 / .0002) It is interesting to compare these values to the "true" conductivities from Table IV.2. We can see excellent estimates for the scalp and skull conductivities and a little bit less accurate estimates for the intracranial tissues.

Finally, in Figure 5.5 we present the dynamics of the performance of the inverse search in our distributed multi-cluster computational environment. Four curves with different markers show the dynamics of the inverse solution flux at the

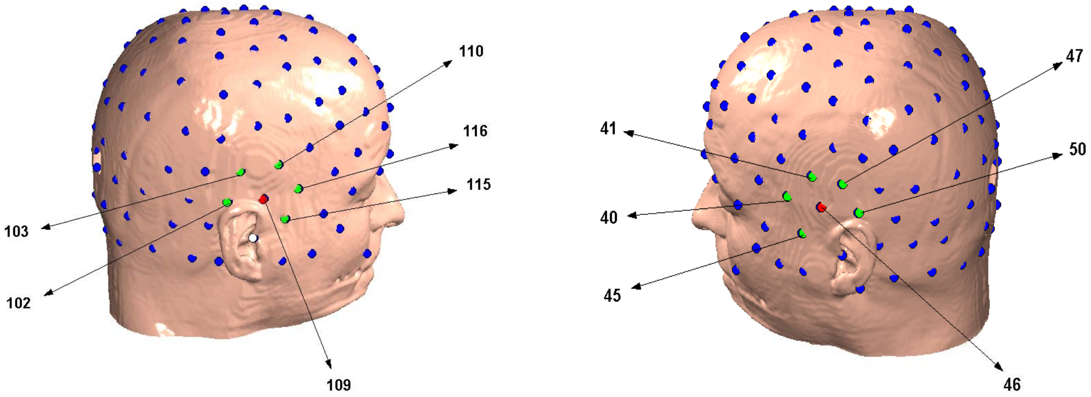


FIGURE 5.6: The electrodes map on the scalp. Electrodes marked with red circles are the current sources. Electrodes marked with green are execluded in the inverse computation as outliers.

master controller. One can see that Clust4 on average returns the inverse solution twice as fast as the other clusters, as would be expected. Note, however, the time to inverse solution also depends on both forward speed and convergence rate. The markers seated at the ”zero” error function line represent solutions that contribute to the final solution distribution, with the rest of the solutions rejected as outliers. In average, the throughput was 12 minutes per one inverse solution for

$128 \times 128 \times 88$ MRI resolution.

Primilinary Results with Real Data

Having the inverse solver tested by simulating the experimental set of potentials using the forward solver with mean conductivity values obtained from the literature, in this section we present the preliminary results using experimental data obtained by a set of electrodes on a human subject. The data was recorded at

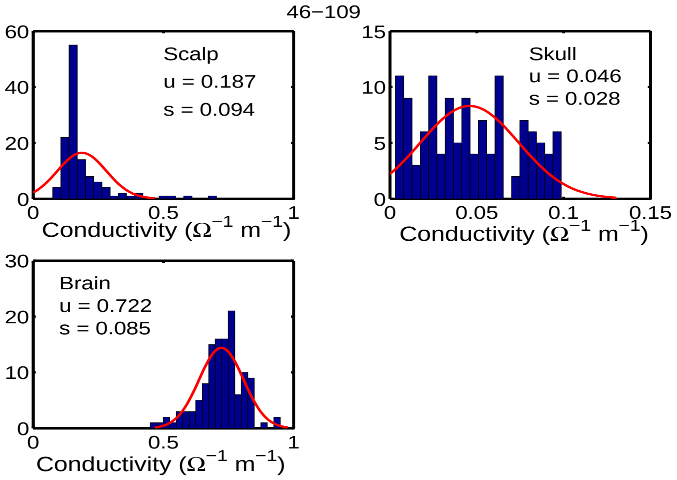


FIGURE 5.7: The distribution of Scalp, Skull and brain conductivities as obtained from the inverse search.

Electrical Geodesics Inc (EGI)[57] using 129-channels net as shown in Figure 5.6 and EGI Photogrammetry System [57]. Two channels (109–46) used for injecting $1\mu A$ current. One channel (129) used as reference and the rest 126 channels simultaneously acquire the response to the injected current. The geometry of the tissues is obtained from a CT scan segmented into three tissues (Scalp, Skull and Brain). In this study we used low-resolution geometry ($2mm$). Figure 5.7 shows the distribution of the retrieved conductivities. As we see in the plot the skull conductivity is less accurate. In Figure 5.8 we show a comparison of the measured potentials at the electrodes with the potentials obtained using the forward solver with the retrieved conductivities. In addition, we have plotted the potentials for the mean conductivities found in the related biomedical literature. As we see in the plot the pattern of the obtained potentials matches the data. On the other hand, the values of the potentials at some electrodes disagree with data. Further, our results

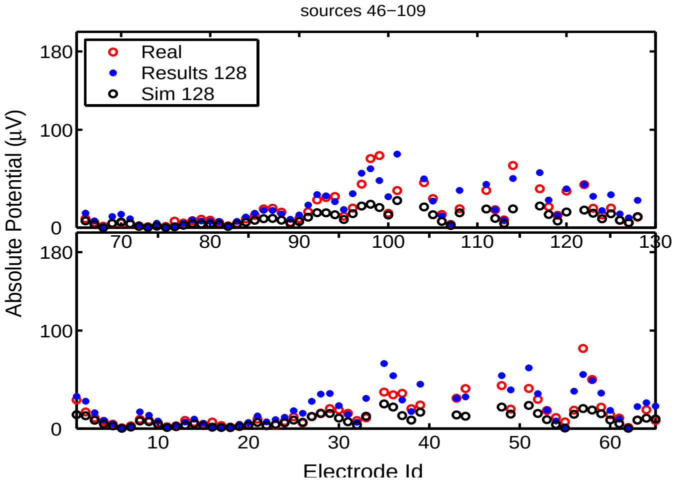


FIGURE 5.8: Comparisons of the measured potentials, the calculated potentials using the average conductivities from literature and the calculated potentials using the retrieved conductivities.

show, the skull/brain conductivities ration is about 15 as has been observed in other study.

Since it is impossible to obtain conductivity values from experimental measurements on a human subject, our criteria to validate the results is by comparing the results obtained using different current injection pairs. We repeated the same analysis for different current injection pairs. The results corresponding to different pairs do not match each other well. The mismatch is expected due to several sources of uncertainties coming from 1) errors in the measured data 2) simplification of modeling the head as a volume conductor (ignoring skull inhomogeneity and other skull variations, ignoring anisotropy, using low resolution geometry, and so on) and 3) the ill-posedness of the problem itself.

5.4.3 Discussion

The aim of conducting the simulation study in this section is to prove the feasibility of our approach. It was believed that this approach is not possible because most of the injected current will be shunted in the scalp. Our simulation results indicate the validity of bEIT method to estimate the subject's own tissue conductivities. However, the accuracy of the results is highly influenced by several factors related to the accuracy of the measured data, the accuracy of the forward solver, and how to overcome the ill-posed nature of the inverse problem. Improving the accuracy of the results can be achieved by improving the accuracy related to these factors as summarized below:

- Improving the accuracy of the measured data by improving the data acquisition system, increasing the amount of injected current, and applying sophisticated signal processing tools to filter the data from the noise.
- Increasing the spatial sampling accuracy by increasing the number of sampling electrodes (e.g., from 128 electrodes to 256 electrodes).
- Increasing the number of current injection pairs, (e.g., from 4 current injection pairs to 60 or more current injection pairs) and then to apply statistical approach to extract the most likely set of conductivities that provide best match to all measured data.

- Improving the forward model, by including anisotropy and inhomogeneity in the model of the anisotropic and inhomogeneous tissues (In particular for the skull tissue as discussed in Section).
- Improving the optimization algorithm for the inverse problem that better find the global minimum and can extract larger number of tissues to be included in forward model which allows modeling more features.
- Incorporating a priori information in the form of constraints to guide the search toward the feasible regions in the inverse problem.

One goal of this dissertation is to provide the computational tools and methods necessary to improve the computational aspects of this problem. However, if we increased the numbers of tissues to more accurately describe the volume conductor such as including inhomogeneity or anisotropy, the convergence rate and probability of the simplex algorithm diminishes quickly. Six tissues are enough to reduce the probability of convergence to less than 10% and of finding optimal solutions to much less. Therefore, a new more robust algorithm that can extract more number of tissues is needed. In section §5.5 we describe a HPC computational environment based on parallel simulated annealing algorithm which overcome the limitations imposed by simplex search and allows us to find the global minimum more accurately for larger number of tissues. In section §5.6 we provide a method to include inhomogeneity into the skull model while keeping the number of tissues tractable.

5.5 Conductivity Modeling - Simulated Annealing

Two factors argue for a new approach. First, experimental studies reported that the skull tissue is anisotropic and highly heterogeneous and cant be modeled as a uniform tissue 4.3.3. Second, if we increased the number of segmented tissues to more accurately model skull inhomogeneities, the convergence rate and probability of the simplex algorithm diminishes quickly 5.4.3 Therefore, we replaced the simplex method with a simulated annealing algorithm that can discover conductivities up to thirteen segmented tissues. We have parallelized the conductivity search problem and evaluated its convergence and scalability attributes on the SDSC DataStar machine. We have also prototyped an alternative parallelization strategy and tested its scalability potential. In this section we describe the computational environment design we used. In §5.5.2 we describe the two-level parallelism design and in §5.5.2, we describe the three-level parallelism.

5.5.1 Simulated Annealing Algorithm

Simulated annealing (SA) is one of the most powerful optimization techniques. It is based on a Monte Carlo simulation that simulates the physical process of annealing to obtain perfect crystals [128]. The original Metropolis [128] simulation was proposed to find the equilibrium configuration of a collection of atoms at a given temperature that minimizes the system energy. Many years later in 82 Kirkpatrick et al [114] was the first to propose simulated annealing as an optimization algorithm to solve combinatorial problems. Simulated annealing can be considered as a modification to the hill climbing algorithm by adding a stochastic decision and a cooling schedule. In hill climbing algorithm, a trial solution is generated from the current solution by mutation and only if it performs better, it replaces the current solution. Similar to hill climbing, in simulated annealing the trial solution is always accepted if it performs better. However, it still can be accepted with some probability if it performs worse than the current solution. This stochastic decision allows the algorithm to escape from the local optima. The acceptance probability is a function of the system temperature and it decreases over time,

$$P = \exp(-\Delta F/T),$$

where ΔF is the increase in the cost function and T is the temperature in the simulation (a parameter that controls the acceptance probability of the worse solution). Initially when T is high the algorithm accepts inferior moves with higher probability. However, as the system cools down the system accept inferior moves with smaller probability. The simulated annealing algorithm consists of three

Algorithm 1: Sequentional simulated annealing algorithm.

```
input  : Initial Temprature $T_0$ and initial Point $X_0$
output: $X_{optimal}$
$T = T_0$, $X = X_0$, $F = Cost(X_0)$
while $T > 0$ do
    for $i \leftarrow 0$ to $N_t$ do
        for $j \leftarrow 0$ to $Ns$ do
            for $k \leftarrow 0$ to $N$ do
                $X_k \leftarrow purturb(X, k)$
                $F_k \leftarrow Cost(X_k)$
                $\Delta F \leftarrow F_k - F$
                if $\Delta F < 0$ then  accept $X_k$ and update $X_{optimal}$
                else  accept $X_k$ with propability $\propto \exp(\Delta F/T)$
        adjust-maximum-step-length
    check-termination-condition
    reduce temparature $T = rT$
```

nested loops 1

Temperature cooling: The outer loop controls the temperature cooling; the temperature is reduced by a factor of r after executing N_t search-radius loops. Many other cooling schedules are proposed in the literature and finding the best cooling schedule is considered one of the disadvantages of the SA. The advantages and disadvantages of SA algorithm are discussed in [106].

Neighborhood Search radius: In combinatorial problems the neighborhood trial solutions of a solution is generated by a small mutation to the current solution. However, in a continuous problem the distance in the search space defines the neighborhood of the current solution. Therefore, in a continuous problem the neighborhood solutions of a solution are those solutions that can be reached within a certain radius in the search space. The size of the search radius determines the characteristics of the search. Large radius causes the search to be explorative and less likely to converge to an accurate solution. On the other hand, small radius causes the search to be exploitive and less likely to explore the search space well. Therefore, normally an adaptive mechanism is used to adjust the search radius such that early in the search, it is large (explorative) and it decreases as the search gets closer to the solution (exploitive). Several approachs are proposed to adjust the search radius adaptively. These include using Gaussian distribution [106] or Cauchy distribution [196]. In these approaches the standard deviation of the distribution is the temperature parameter. Therefore the search radius is large when temperature is high and the radius decrease as the temperature decreases. Another approach, which we used in our implementation, is proposed by Corana's [39]. In this approach the search radius is adjusted based on the rate of the accepted to the rejected moves. The advantage of Corana's approaches is that the objective function is considered in the adjustment. Based on this algorithm, the trial solutions x_0 is generated from the current solution x as follows,

$$x'_i = x_i + rand() * V_i,$$

where $rand()$ is a random number between 0 and 1, and V_i is the search radius in direction i. V_i is adjusted at the end of the neighborhood search loop as follows,

$$V'_i = V_i * g(p),$$

$$g(p) = \begin{cases} 1 + c\frac{p-0.6}{0.4} & \text{if } p > 0.6 \\ \left(1 + c\frac{0.4-p}{0.4}\right)^{-1} & \text{if } p < 0.4 \\ 1 & \text{otherwise} \end{cases},$$

where c is an adjustment parameter (suggested to be 2), p is the ratio of the accepted moves to the rejected moves. When the number of accepted moves increases the search radius will increase and when the number of rejected moves increase the search radius will decrease. The above setting is to keep the ratio of the number of accepted moves to the number of rejected moves between .4 and .6.

Control point: The inner loop considers new moves in all directions by perturbing the control point in all direction. The perturbation is constraint between 0 and the maximum step length in each direction.

Transitions that lower the cost function are always accepted, while transitions that raise the cost function are accepted with probability based on the temperature and the size (cost) of the move (Metropolis criteria). The acceptance of the uphill moves allows the system to overcome local minima and make the algorithm insensitive to the initial starting values. The simulated annealing converges when the minimum become stable that does not change for more than epsilon after several temperature reduction loop iterations. The complete algorithm is given in Algorithm 1. The choice of suitable initial temperature, search radius and the cooling schedule are highly problem dependent and normally are chosen empirically and they have a significant impact on the performance of the algorithm. A short summery of some of the effort to guide the parameter selection is discussed in [172].

5.5.2 Parallelizing Simulated Annealing

One of the main disadvantages of simulated annealing algorithm is its performance in finding the global optima like other global optimization algorithms. Therefore, many approaches are proposed to parallelize it. In this section we review some of this effort, which is mainly focused on continuous problems. As discussed before the sequential simulated annealing starts the search at high temperature and reduces it slowly until the system is frozen. At each temperature, a local search in the neighborhood of the current solution is performed until equilibrium. The range of the neighborhood is adjusted adaptively. A simple approach to parallelize SA is to decompose the domain into subspaces. Then a sequential SA instance is assigned to each subspace. However, this approach is not suitable for continuous problem as each subspace still has infinite possible solution. Even in combinatorial problems, it

Algorithm 2: Parallel simulated annealing algorithm on the intermediate level

```
input  : Initial Temprature $T_0$ and initial Point $X_0$
output: $X_{optimal}$
$T = T_0$, $X = X_0$, $F = Cost(X_0)$
$N_{task} = N_s$/Number of tasks
while $T$¿0 do
    for $i \leftarrow 0$ to $N_t$ do
        $X_{best} = X$, $F_{bset} = F$
        for $j \leftarrow 0$ to $N_{task}$ do
            for $k \leftarrow 0$ to $N$ do
                $X_k \leftarrow purturb(X, k)$
                $F_k \leftarrow Cost(X_k)$
                $\Delta F \leftarrow F_k - F$
                if $\Delta F < 0$ then accept $X_k$ and update $X_{best}$
                else accept $X_k$ with propability $\propto \exp(\Delta F/T)$
        MPI communication
        adjust-maximum-step-length
    Master task gets each task best point
    Update $X_{optimal}$
    check-termination-condition
    Master task Brodcast $X_{optimal}$
    $X = X_{optimal}$, $F = F_{optimal}$
    reduce temparature $T = rT$
```

is not always possible to decompose the domain, consider for instance a scheduling problem. There are two general approaches to parallelize SA, a *single-move acceleration* approach and a *parallel moves approach.* In single-move acceleration approach a trial solution is generated and the objective function at this solution is evaluated in parallel. This approach can be efficient if the objective function is costly and can be parallelized efficiently. However, the degree of parallelism will be limited to the objective function parallelism. In the parallel moves approach or multiple-trial approach, several trial points are generated and evaluated in parallel. Further, we discuss our new algorithm to parallelize the inner loop.

In our research work we combined these approaches in a hieratical computational environment and in Chapter VI we show how ODESSI framework supports these parallelism for arbitrary simulation without any effort on the scientist part. In the following we discuss two parallel moves approaches. The first one is based on parallelizing the temperature loop while the second is based on parallelizing the search radius loop. Also, we discuss our new algorithm to parallelize the inner loop.

Temperature Parallel Simulated Annealing Temperature parallel simulated Annealing (TPSA) proposed by Kimura and Taki [111] to solve combentorial problems. The motivation behind this approach is to overcome the low performance that can result due to bad temperature scheduling. In this approach the algorithm can run in parallel without a need for temperature scheduler. Miki et al [129] extended TPSA algorithm to continues SA algorithm and they uses adaptive neighborhood search radius. In TPSA, processors are assigned different temperatures T_p and each processor executes a sequential SA using its assigned temperature. Then two processor having adjacent temperatures exchange their solutions with certain probability at intervals of the annealing time as follows:

1. If the solution at higher temperature is better than the one at lower temperature, the solutions are exchanged.
2. Otherwise, the solutions are exchanged with a probability that depends on the difference in the temperature and the difference in the cost function.

$$P(T,E,T',E') = \begin{cases} 1 & \text{if } \Delta T \dot{\Delta} E < 0 \\ \exp -\frac{\Delta T \dot{\Delta} E}{TT'} & \text{otherwise} \end{cases} .$$

The idea is to perform the search at lower temperature for better solutions to guide the search around good solutions and to perform the search for inferior solutions at higher temperature (more explorative).

Concurrent neighborhood search In this approach discussed in [172], the processors cooperate in the search of the neighborhood of the current solution. Starting with the same initial solution, all processors start a search in the neighborhood of the current solution independently. When a worker accepts a new solution it reports to the master. The master waits the other workers to complete their current trials and then updates all workers with the best-accepted solution obtained among all workers. This approach is classified as a single markov chain SA because all workers are working on the same chain (a single move is selected from a number of moves reported by the workers). In [109] they used this approach and we used their approach in our implementation as discussed bellow.

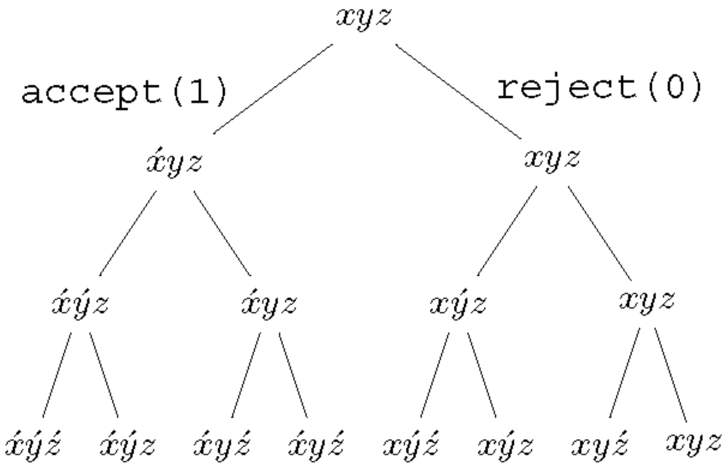


FIGURE 5.9: SA-inner loop parallelism, path selection.

Inner loop parallelism To create greater potential parallelism, it is possible to enumerate all possible perturbation paths around the control point, increasing the number of candidate points from N to 2^N, and breaking the serialization introduced in the inner loop when the next perturbation point depends on the acceptance or rejection of the prior point. This new algorithm effectively multiplies the degree of parallelism by $2^N/N$ with a potential performance gain of N. In this algorithm, the simulated annealing criteria in selecting the path are preserved (we just computed all possible paths concurrently). Therefore, this algorithm guarantees to produce at least as optimal results as the former, but it requires significantly more resources to achieve greater performance.

If N is the number of variables, then there are 2^N distinct possible paths, and the possible path ranks are $1 - (2^N - 1)$. Each path can be described by the binary representation of its rank. For example, for three tissues we have 8 possible paths which can be specified by their binary representation $(001, 010, \ldots, 111)$. The position of a bit in a path corresponds to the direction at which the control point is perturbed and its value corresponds to whether that move is accepted or not. For example, assuming the control point is $P = (x, y, z)$ the path (101) corresponds to the following sequence a) The first bit from the left (1) corresponds to P_1 is generated by perturbing P in the x-direction and the move is accepted, so the control point P becomes P_1 b) the second bit from the left means P_2 is generated by perturbing $P = P_1$ in the y-direction and the move is rejected, so the control point remains $P = P_1$ c) the third bit from the left (1) corresponds to P_3 is generated by perturbing $P = P_1$ in z-direction and the move is accepted, so the control point P becomes P_3.

Then all possible candidate points (2^N) are generated and the objective function values at these points are evaluated in parallel. To compare between the solution obtained by this algorithm with the original algorithm, we used N random number $[r_1, \ldots, r_N]$ such that r_1 is used to perturb in 0 direction and so on. Next we need to select the path that preserves the simulated annealing Metropolis criteria, the reject/accept depends on the difference of the cost function between the control point P and the candidate new point, also it depends on the Temperature. We implemented this by building a tree as shown in the Figure 5.9 for three tissues example. The master task constructs the tree with all possible paths (the cost

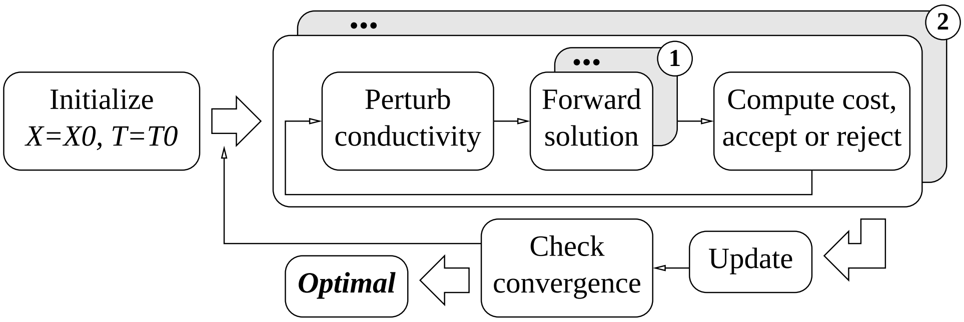


FIGURE 5.10: Two level parallelisms: A parallel simulated annealing algorithm on the level of the intermediate loop, each task runs the forward calculations in parallel.

function is computed in advance at all these nodes). Then the simulated annealing criteria is applied to the left child node if not accepted we go right. The selection algorithm is shown in Figure 3.

Algorithm 3: SA inner loop path selection.

```
input  : Control Point $P$
output : New Control Point $P$, $P_{best}$
for $k \leftarrow 0$ to $N$ do
    $P_{try} = P \rightarrow left$
    if $accept(P_{try})$ then
        $P = P \rightarrow left$
        update $P_{best}$
    else
        $P = P \rightarrow right$
```

5.5.3 Computational Design

As discussed before §5.5, several tissues must be modeled to describe an accurate volume conductor (e.g., inclduing inhomogeneity and anistropy). However, increasing the number of tissues, increases the computational complexity. Our interest is to determine the tradeoffs between number of tissues, simulation time-to-solution, and conductivity model accuracy. For this purpose, we computed the scalp potentials for thirteen preset tissue conductivities and tested how well the model performed as the number of tissues varies. To address the increased computational demands, we parallelized the conductivity search. In this section we describe the computational design we adapted based on the two- and three-levels parallel simulated annealing algorithms described in sections §5.5.2 and §5.5.2.

Since the forward solution is the highest cost component of the conductivity and source modeling, we parallelized the forward solver using OpenMP §4.5.3. Here, we also parallelized the simulated annealing algorithm along the search radius, based on the MPI-based methods described in [109] . Figure 5.10 shows a high-level view of the parallel simulated annealing approach, with the intermediate loop distributed across several nodes (1 task per node), each running parallelized forward calculations. All tasks start with the same initial values, but with different random generator seeds. Each task perform a random search around the control point by perturbing the control point in all direction. The perturbation is constrained to be within the maximum step length. At the end of the search radius loop, the master task gathers every task best solution and updates the optimal solution and all nodes communicate to adjust the maximum step length. At the end of the temperature reduction loop all tasks updates the control point with the optimal solution.

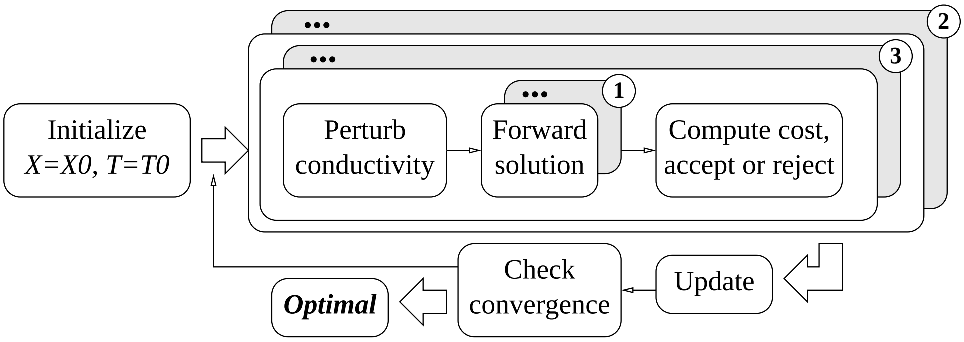


FIGURE 5.11: Three level parallelisms: In addition to the parallel forward calculation, each task computes the inner loop in parallel.

Most of the performance numbers reported in the next section are for the parallel-simulated annealing algorithm in Figure 5.10. Choosing twelve tasks and 16-way forward solves allows the amount of parallel execution to reach 192 processors. There does not appear to be much benefit in increasing the number of tasks beyond twelve and the OpenMP performance flattens beyond sixteen processors. To create greater potential parallelism, we decided to enumerate all possible perturbation paths around the control point as described in §5.5.2. This new algorithm (shown graphically in Figure 5.11) effectively multiplies the degree of parallelism by $2^N/N$ with a potential performance gain of N. The numbers in Figure 5.11 for each case indicate the different types of parallelism. Interestingly, the nature by which this algorithm was created guarantees to produce at least as optimal results as the former, but it requires significantly more resources to achieve greater performance benefits.

5.5.4 Results

We conducted a series of experiments to test both the convergence properties and the performance of the conductivity modeling based on parallel simulated annealing. All experiments were performed on the San Diego Supercomputing Centere DataStar system [185], a cluster of shared-memory nodes consisting of 16-processor IBM p690 and 8-processor IBM p655 machines. All results presented below were performed using 8-way OpenMP tasks running forward calculations each on a separate p655 node.

As mentioned earlier, we preset thirteen tissue conductivity values (eleven skull parts, scalp, brain) and ran experiments to test the conductivity model accuracy on fewer tissue numbers. We started with eleven tissue to verify convergence to acceptable values. The simulated annealing search starts with initial random conductivities selected from the biomedical ranges and stops when one of three criteria is met as described in the computational design. Our results verify the ability of simulated annealing to extract eleven tissues with good accuracy and precision. Figure 5.12(left) shows the dynamics of the 11-tissue inverse search convergence, giving the temperature cooling, the cost function, and one tissue's conductivity. This calculation was done on a single 8-processor p655 node in our lab and took approximately 31 hours to complete.

Having verified convergence for a large number of tissues, we decided to limit the number of tissues to five (three skull parts, scalp, brain) to test the performance

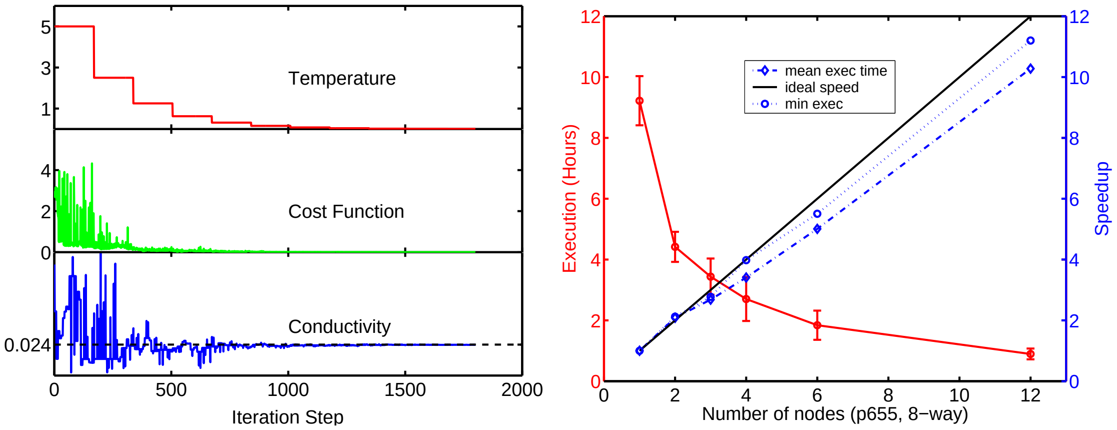


FIGURE 5.12: Solution flow at the master controller. Inverse solution arrival to the controller are marked.

properties of the parallel simulated annealing algorithm. This will allow us also to contrast performance with the earlier simplex outcome. The execution time and performance speedup for a $2mm$ resolution problem from one to twelve tasks (each task run on a 8-processor p655 node) is shown in Figure 5.12(right). The speedup is almost linear with the number of nodes. Three experiments were run on each data point to show the performance variation due to the random number generator sequence.

Even though we have excellent speedup, the degree of parallelism is limited. The second version significantly increases parallelism by generating N random numbers, one for perturbing the control point in each direction, and generated all possible points by enumerating the perturbation in all directions. Then the cost function at these points is evaluated in parallel. After computing all possible paths, the simulated annealing criteria is applied. In theory this parallelism can speedup the computation by a factor of N. In addition, we can get further speedup by selecting the best point from all points that was computed and not only from those points on a simulated annealing path. This speedup is due to speeding the convergence. For verification purposes, we compared this new algorithm with the former for a problem where the conductivities of three tissues are found. We used a single task and two processors for the forward calculation. Thus, in the former algorithm, the parallelism degree is two. In the new algorithm, the parallelism is of degree sixteen (2^3 perturbed points by two processors in the forward calculation), allowing two 8-processor p655 nodes to be used, with a speedup potential of three (the number of tissues). Our experimental results show an overall performance improvement of 1.98. The best point selection gives a convergence speedup of 8%, while the inner loop parallelism produces a speedup of 1.77. We believe better speedup can be achieved by eliminating communication overheads. For larger numbers of tissues, the number of processors needed to realize the potential parallelism in the new algorithm increases by a power of two. When this resource scaling is unavailable, the method can be throttled to use a smaller degree of perturbation fan out.

Effect of Skull Inhomogeneity

Having evaluated the robustness and scalability of the simulated annealing based computational environment, we were able to start conducting scientific

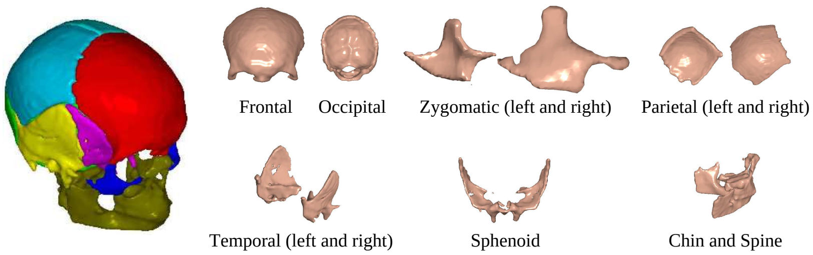

Frontal
Occipital
Zygomatic (left and right)
Parietal (left and right)
Temporal (left and right)
Sphenoid
Chin and Spine

FIGURE 5.13: Anatomically parcellated-skull into 11 major bones.

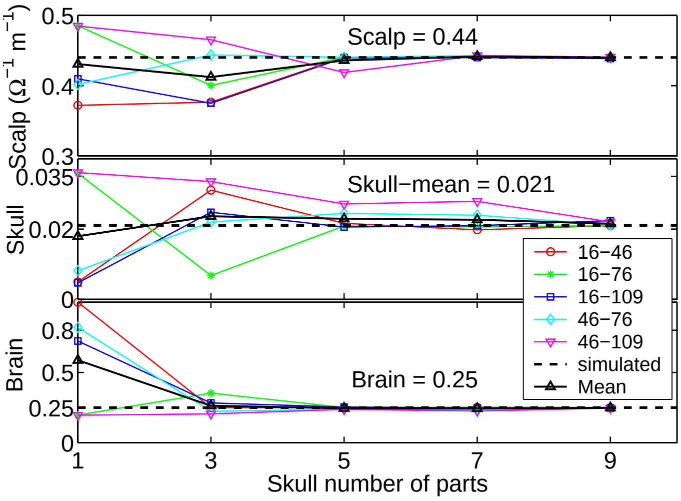


FIGURE 5.14: Brain, Scalp and skull-mean conductivities as a function of the assumed number of skull parts in the inverse search, the simulated data is generated using 11-part skull.

investigations to better understand the characteristics of the head modeling. The first investigation we conducted is to study the effect of skull inhomogeneity on the extracted conductivities. To do so, we generated the simulated data using 11-parts anatomically based parcellation as shown in Figure 5.13. Each part represents a major skull bone. We parcellated the skull by implementing a Matlab tool that allows us to interactively chop parts of the skull. We considered 8 cranial bones and

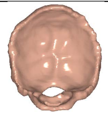
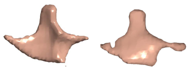
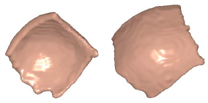
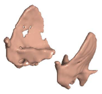
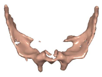
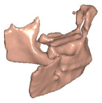
						
Frontal	Occipital	Zygomatic	Parital	Temporal	sphenoid	Chen&spine
(1) 22077 0.004	(2) 17470 0.008	(L(3)&R(4)) 1654 0.012,0.016	(L(5)&R(6)) 23532 0.02, 0.024	(L(7)&R(8)) 15099 0.028,0.032	(9) 5589 0.036	(10)&(11) 17984 0.04, 0.044

TABLE V.1: Anatomy-based parcellation (size in voxels and conductivity in S/m.

3 facial bones. The conductivity of each part is assumed to be homogeneous. The bones and their sizes are shown in Table V.1.

Using the forward solver, we generated five different simulated measured-data sets correspond to five differnt current injection pairs by assigning a conductivity value (shown in Table V.1) for each part. The skull weighted-mean conductivity is .021 S and the skull weighted-standard deviation is .0147 S. Then using the inverse solver, we retrieved the conductivities assuming the skull is uniform, 3, 7, and 9-parts. The assumed 3-parts skull was formed from bones $\{(2,6,7), (3,11,12), (4,5,8,9,10)\}$, the 5-parts skull from bones $\{2,3,(4,5,8,9,10), (6,7), (11,12)\}$, 7-parts $\{2,3,(4,5), (6,7), (8,9), 10, (11,12)\}$ and 9-parts $\{ 2,3,(4,5), 6,7,8,9,10, (11,12)\}$. Figure 5.14 shows the extracted brain, scalp and skull weighted-mean conductivities (the weights are the part's sizes) using 5 different current injection pairs as a function of skull number of parts. As we see in the figure the extracted conductivities approach the simulated ones as we parcellate the skull. The red circles line in the figure is the average tissue conductivity using all current injections pairs. The skull weighted-mean conductivity is computed using the following formula,

$$\sigma_{skull} = \frac{1}{\sum_{i=1}^{m} s_i} \sum_{i=1}^{m} \sigma_i \times s_i, \tag{V.5}$$

where s_i and σ_i are the size of the skull part i (in voxels) and its extracted conductivity (in Semens), and m is total number of parts. The error in the extracted conductivity is calculated using the following formula,

$$ER = \frac{1}{P} \sum_{i=1}^{P} \frac{|\sigma_{retrieved} - \sigma_{simulated}|}{\sigma_{simulated}}.$$

Error up to 140% can be produced assuming skull is uniform for a synthetic data generated assuming a skull conductivity variation of 70% (Coefficient Of Variation = stdev/mean) and shown in Figure 5.14. Also, as we see in the figure, the error caused by assuming the skull is uniform depends on the location of the current injection pairs relative to the skull parts.

5.6 Skull Inhomogeneity

The most important factor in head modeling is to obtain an accurate conductivity model of the skull. Published data about the skull conductivity values cover a wide range and highly controversial as summarized in table IV.1. These large variations in the skull conductivity values expose the complexity of the skull

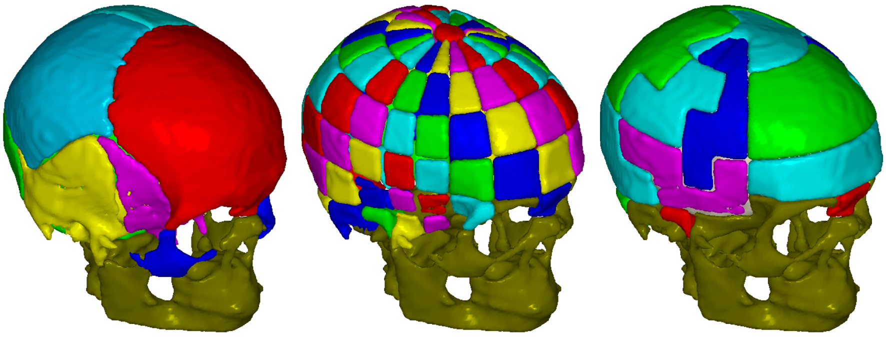


FIGURE 5.15: Skull parcellation: a) An 11-parts anatomy-based parcellation of the skull b) A 97-parts thickness-based parcellation c) A 5-parts thickness-based parcellation formed by combining the parts in b.

structure. However, these studies provide a valuable amount of information. For instance, all measurement studies agree on several general observations about the skull: 1) sutures are highly conductive 2) the spongy middle layer of a trilayer bone is more conductive compared to the outer two layers 3) the inner layer of the trilayer bone is more conductive than the outer layer 4) the conductivity of the trilayer is related to its thickness. These common general observations about the skull can be used to construct more accurate skull conductivity model. Combined with accurate geometrical information, the use of EIT scalp current injection, and advanced numerical solvers will provide opportunity to create an accurate model of the head volume conductor. In this section we formulate a method to include skull inhomogeneity and we incorporated a priori knowledge from experimental studies to reduce the necessary number of parameters needed in the inverse search.

It is natural to assume that anatomically different parts of the skull have different conductivity values, and experiments bear this out §4.3.3. Thus, it is

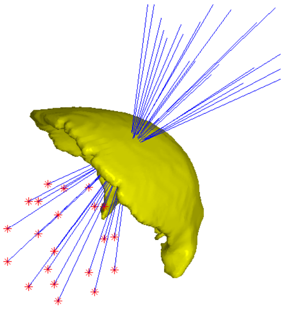


FIGURE 5.16: Thickness computation: The thickness at point p located on the inner surface of the skull is the smallest thickness computed by rays casted from points located on the surface of the sphere.

important to characterize the skull inhomogeneities as much as possible in the conductivity modeling. However, increasing the number of modeled tissues also increases search and computational complexity. Our interests are to determine the tradeoffs of tissue dimensionality, simulation time-to-solution, and conductivity model accuracy.

Our approach to include skull inhomogeneity, is to anatomically parcellate the skull into three different type tissues: *trilayer* bones, *compact* bones and *sutures*. The compact bones can be further parcellated based on their anatomical properties, and their location relative to the electrodes. Sutures and compact bones parcels are treated as seperate tissues in the inverse solver. For the trilayer bones, we used the linearity relationship between thickness and conductivity to obtain the inhomogeneity in the conductivity from the inhomogeneity in thickness. The inhomogeneity in thickness can be obtained from the imaging modality such as MRI. Therefore, variations in conductivity is captured from the variation in thickness.

Mathematically, the conductivity at point r in the trilayer bones is given by,

$$\sigma_{trilayer}(r) = A/t_{trilayer}(r), \tag{V.6}$$

where $\sigma_{trilayer}, t_{trilayer}$ are the trilayer bones conductivity and thickness at point r, and A is the proportionality constant. In principle we need to compute the thickness at every point on the skull trilayer bones surface. To simplify the computation, we approximated the thickness at each point by the mean thickness of a trilayer parcel that contains the point. We accomplished this by parcelating the trilayer bones of the skull into N parcels. Then we computed the mean thickness of each parcel. The conductivity of each trilayer bone parcel is given by,

$$\sigma^i_{trilayer} = A/t^i_{trilayer}, \tag{V.7}$$

where $\sigma^i_{trilayer}$ is the conductivity of a trilayer parcel i and $t^i_{trilayer}$ is its mean thickness. The parcel mean thickness is computed by averaging the thickness at several points uniformly distributed on the parcel inner surface. As N gets larger the thickness gets closer to the continuous values. The goal of the inverse search is to find 1) the proportionality constant parameter A 2) the conductivities of the compact bones parcels 3) the conductivity of the sutures parcels and 4) the conductivities of the other tissues (brain, scalp). In other word, the predicted potentials on the scalp Equation V.2 becomes,

$$\phi^p = F(\sigma_{brain}, \sigma_{scalp}, \sigma_{suture}, \sigma_{compact}, \sigma_{trilayer}^{i=1...M}), \tag{V.8}$$

where M is the number of parcels in the trilayers bones. The conductivity of the trilayer bones is computed using the proportionality constant A and the thickness. The thickness can be obtained from the imaging modality. This means that the inverse search needs to find only one parametr (the constant A) to determine the conductivites of the trilayer bones. When the inverse solver varies the constant A, it first computes the conductivities of trilayer bones parcels using Equation V.7 , and then the predicted potential on the scalp is computed using Equation V.8.

We investigated this method on the cranial part of realistic skull obtained from CT scan. First, we parcellated the cranial part of the skull into N parts as shown in Figure 5.15(b). Out of the N-parts we constructed m-parts parcellations (where $m < N$) by distributing the N-parts into m-bins thickness histogram, and then we reunite the parts that fall into each bin with a given thickness value to map it back to the skull and form a new m-parts parcellation pattern. Figure 5.15(c) shows a 5-parts parcellation.

Skull Thickness Estimation We computed the skull thickness at point p located on the inner surface of the skull by casting several rays from points distributed uniformly on a surface of a sphere as shown in Figure 5.16. The sphere is centered at the center of the head with a radius smaller than the radius of the skull. After several trials we chose the radius to be about .25 the radius of the skull.

Each ray enters the skull at the same point p and leaves the skull at some point p_{ray} on the outer surface of the skull. The thickness t_{ray} at point p computed using each ray is the geometrical distance between p and p_{ray}. Then the thickness at point p is the smallest thickness obtained by all rays. The idea behind this approach is to explore several angles in penetrating the skull at point p . We verified this approach by manually computing the thickness at several points.

5.6.1 Skull Inhomogeneity Results

To reduce the number of parameters in the inverse search while including the inhomogeneity in the skull model, we considered the observed linear relation between thickness and conductivity for the trilayer bones. We restricted our study on the cranial part of the skull. The conductivity of the facial bones is fixed at .018 S/m through out this study. The reason is that it is hard to define and compute the thickness of these bones and facial bones are mostly compact bones that the relation between thickness and conductivity doesn't apply. Additionally, their impact on EEG is less than cranial plates. We note here that assuming the linearity relation between the conductivity of the cranial bones and their thickness is equivalent to introduction of a priory information (which comes from the experiment) and effectively reduction of the unknowns, in this case all conductivity carrying information is presented by the coefficient of proportionality A, while the rest is given by the local geometry (thickness).

We investigated the effect of thickness accuracy on the extracted tissues using the inverse solver. We first approximated the skull thickness inhomogeneities by parcellating the skull cranial part into 97 parts. The mean thickness of each part is computed by averaging the thickness at about 20 points uniformly distributed through the part. We generated the synthetic data (simulated measured data) such that the linearity relation between thickness and conductivity is applicable for the trilayer bones by assigning for each part i of the trilayer parts a conductivity of $\sigma_i = A/t_i$, we chose $A = 1.234Sm$. The facial part of the skull is assumed to be uniform with conductivity $.018S$. The brain and scalp conductivities are pre-set to $.25S$ and $.44S$. Then using the inverse solver we extracted the conductivities of the brain, scalp, facial bones, and the constant A assuming m-parts parcellation for $m = 1, 2, 3, 5, 6$ As we see in the figure, the extracted conductivities and proportionality constant A are closer to the preset values as we consider parcellations with larger number of parts effectively smaller deviation in thickness. Table 5.6.1 showes the extracted conductivities and their error and the standard deviation in computing the average thickness.

5.7 Summary and Conclusions

In this chapter we introduced a method called bound Electrical Impedance Tomography (bEIT) for reconstructing the conductivities of the human head tissues and we provided an efficient computational environment that proved the feasibility

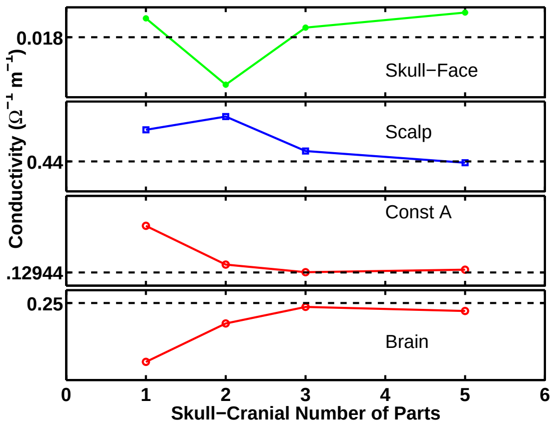


FIGURE 5.17: Conductivity reconstruction: Retrieved conductivities and the parameter A, using m-parts skull (m=1–5).

	Synth.	**Retr.** (96)	**Retr.** (4)	**Retr.** (5)	**Retr.** (6)	**Retr.** (8)
Brain	0.25	0.249	0.227	0.242	0.248	0.247
Scalp	0.44	0.439	0.461	0.470	0.447	0.439
Face	0.018	0.0180	0.0192	0.0148	0.0186	0.0196
A	0.1284	0.1295	0.233	0.147	0.180	0.136
Cond error			.2551	0.1056	0.0405	0.0173
Thick error			.4181	0.374	0.34	0.3238

TABLE V.2: Extracted conductivities and the parameter A.

of the bEIT approach. The simulated annealing algorithm proved to be robust and stable in extracting up to 13 tissues without any sign of failure. In all the testing we performed in this chapter, the simulated annealing runs never failed to converge.

Also, we conducted a sensitivity analysis to study the influence of the skull inhomogeneity on the accuracy of the extracted conductivities. The results indicate that ignoring skull inhomogeneity and other skull variation can introduce an order of magnitude error. Further, we introduced a method to include skull inhomogeneity while keeping the number of parameters in the inverse search tractable.

However, when working with real data several other factors have influence on the robustness and accuracy of the solution. These factors are related to the accuracy of the measured data, the accuracy of the forward model and the ill-posed nature of the inverse problem. Understanding and quantifying the contribution of error from each source is necessary to prioritize the research and weight these factors according to their contribution to the error.

This kind of analysis requires solving the forward model many times. In Chapter VI we provide the design for a generic HPC framework that make conducting this investigation easy. For instance, in this framework the parallel-simulated algorithm can be run on local machine that uses available distributed resource to provide solutions to the optimizer without the need to run the simulated annealing itself on large clusters. The framework factors out the simulation execution and the objective function evaluation from the optimization method itself. This separation enables testing with different algorithms such as Genetic algorithms without altering the objective function or the simulation execution interface. The framework also, allows conducting sensitivity analysis and experimenting with different forward models.

CHAPTER VI

ODESSI DESIGN

In scientific domains where discovery is driven by simulation modeling there are found common methodologies and procedures applied for scientific investigation. ODESSI (Open Domain-extensible Environment for Simulation-based Scientific Investigation) is an environment to facilitate the representation and automatic conduction of scientific studies by capturing common methods for experimentation, analysis, and evaluation used in simulation science. Specific methods ODESSI will support include *parameter studies*, *optimization*, *uncertainty quantification*, and *sensitivity analysis*. By making these methods accessible in a programmable framework, ODESSI can be used to capture and run domain-specific investigations.

6.1 Introduction

Computational science is now accepted as an important approach for scientific investigation, broadly considered equivalent in its discovery power to theoretical and experimental science. It is typically conducted through mathematical modeling and scientific simulation, leveraging access to advanced, high-performance computers (HPC) to run computational experiments (simulations) that seek to model reality in various domains. The evolution of computational science reflects both a growing need for computational power and increased sophistication of simulation methodology. Early concerns were on access to sufficient HPC resources, motivating research in parallel computing, computational grids, and large-scale storage. More recent research work in computational portals and workflows attempts to simplify resource access as well as provide programming support for coordinating simulation and analysis tasks. With computational horsepower becoming more ubiquitous, there is now growing interest in enhancing the discovery process of scientific investigations. In general, how productivity in computational-based science can be improved in practice will depend greatly on software environments that raise the level of investigation creation, execution, and management.

In scientific domains where discovery is driven by simulation there are common methodologies and procedures. An environment that can capture the shared standard practices and support their reuse across domains could improve productivity in scientific investigation creation and application. Methods such as parameter studies and tuning, optimization, uncertainty and sensitivity analysis, are generally used across simulation fields. Application of these methods in simulation studies typically require executing the simulation many times with different input parameter sets and data files. The environment could capture the common scientific methods in modules that can be contextualized for domain-specific use. The modules would hide the details of backend execution (implemented by the environment infrastructure), while providing an interface for their programming as part of an investigation workflow. The environment could also support other aspects of scientific investigations, including the management of input and output data, the specification of parameters, the post-processsing of results, and the generation of reports. The benefit is to provide a high level of service and automation to the computational scientist to enhance their work throughput and management.

In this chapter we describe our research work to create and apply an environment for supporting scientific investigation called ODESSI (Open Domain-extensible Environment for Simulation-based Scientific Investigation, pronounced “Odyssey”). The environment will facilitate the representation and automation of scientific studies by capturing shared methods for experimentation, analysis, and evaluation used in simulation science in a framework that can be programmed and specialized for domain investigations. ODESSI will be evaluated and demonstrated for scientific studies in the neuroscience domain involving computational modeling of the electromagneticproperties of the human head.

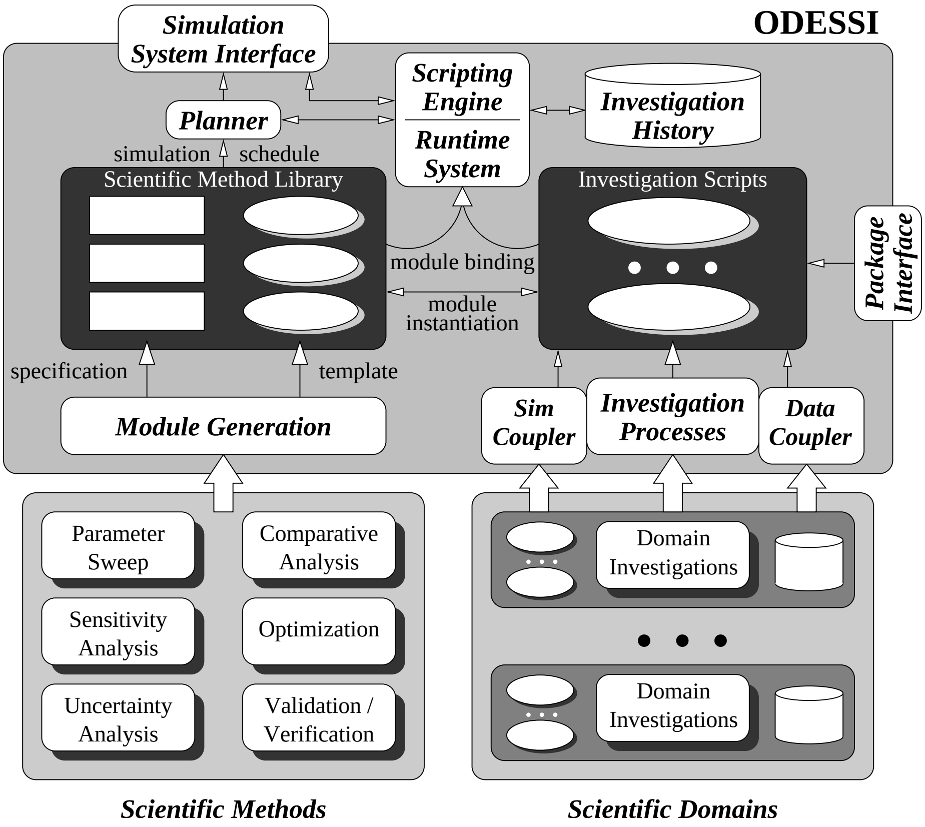


FIGURE 6.1: Architecture and components of ODESSI framework.

6.2 ODESSI Requirements and Design

The goal of ODESSI is to provide a productive environment that assists domain scientists in the development and application of their computational investigations. To this end, the main requirements are:

1. Support common types of scientific methods that occur in several domains and provide a means to apply the methods in a scientific investigation.
2. Provide a programming framework that allows methods and investigations to be developed and applied. The scientific methods should be realized in such a

modular manner to allow method extension and reuse. The scientific investigation should be programmable in a flexible manner.

3. Enable access to high performance computing for purposes of productive simulation studies, while abstracting and hide the complexity of the underlying interactions with the compute resources. The goal here is to insulate the scientist from concerns of HPC resource usage, instead allowing them to focus on the process aspects of the domain investigation.
4. Provide support for persistent and evolving scientific investigations.

The ODESSI environment shown in Figure 6.1 was designed to support these requirements. The key concept of the ODESSI approach is the capture of standard procedures to conduct and analyze (simulation-based) scientific experiments in a modular, extensible, and reusable form. We call these procedures *scientific methods* and think of the methods as generating a set of simulation experiments to run. Common scientific methods include parameter studies, comparative analysis, optimization, sensitivity analysis, and uncertainty analysis. These methods are the basis upon which activities such as verification and validation, parameter tuning, and simulation-based experimentation are built for domain application. These processes that integrate different methods are the foundation of domain scientific investigations. A *scientific investigation* is a domain-specific discovery process that applies one or more scientific methods in its lifetime. It defines the simulation codes to use, the input data files, and post-simulation analysis and visualization. If ODESSI can capture key scientific methods in easy-to-use modules, the level of productivity in the development and execution of scientific investigations may increase. We will focus our discussion on this aspect of the design.

Logically, ODESSI represents methods internally as *modules* consisting of two parts: a *specification* and a *template.* The specification identifies the context necessary for the execution of the modules, including the simulation program to be run and parameters. The template is the software construction of the module with abstract classes for operation of the specific scientific method. In this respect, the template embodies the method procedures for the generation of domain simulation experiments. A module is instantiated by an investigation script, setting the specification context and initializing the module state. When a method module is executed, it generates an experiment schedule (static or dynamic) that is passed to the ODESSI planner.

It is the responsibility of the ODESSI *planner* to conduct the necessary simulations on behalf of the invoked method. It is possible multiple methods are concurrently active, each with its own planner. The planner interfaces with the external simulation system to run a simulation experiment. It determines which experiments to execute based on the specified simulation schedule. If a method uses information from earlier experiments to determine future experiments, its module uses a dynamic schedule which is applied within the planner. The planner attempts to optimize schedules by interrogating the ODESSI *investigation history* to determine when simulation experiments have previously been conducted. A record is maintained in the ODESSI investigation history of every completed simulation experiment, containing complete metadata for the investigation and method specification.

Below, we discuss ODESSI main components. From the user perspective, ODESSI is able to run a large number of simulations that form a domain scientific investigation. Domain scientific investigation is organized from one or more scientific investigation methods in addition to input data files, statistical and visualization tools.

ODESSI architecture consists of following main components (refer to figure 6.1):

Front end. Using the front end, a user can program a domain investigation using ODESSI scientific methods and other external packages by providing a domain investigation script. Each scientific method will have a schema that describes the strategy or the algorithm of the investigation method. The user will be able to specify how the scientific methods are to be instantiated, including how simulation programs and data are to be incorporated. The front end allow the user to interact with ODESSI through the following functionality:

- *Investigation scripting* is the means by which the scientist programs ODESSI to perform a domain investigation by providing a domain script. We use Python as the scripting language.
- The *data coupler* interface allows the user to provide domain data files to be stored in the investigation database and used in the investigations.
- The *simulation coupler* interface allows the user to describe the simulation programs to be used, for instance, the name and path of the simulation executable, the input and output argument names, their data types, and any environment variables. In addtion, the simulation coupler captures the host names and passwords for running the simulation programs.
- *Scientific investigation methods* are input through the front end. They are specified using a schema that can be used to generate a method module in the ODESSI method library.

Package interface. ODESSI support interfaces with other packages needed to perform results analysis (e.g., Matlab, R, SciPy), data mining (e.g., R, WEKA), or visualization (e.g., JFreeChart, MayaVi). These packages can also be used in monitoring the execution of the experiments.

Simulation System interface. ODESSI interface with the computing resources through the simulation manager component. This interface allows ODESSI to run and control the execution of several instances of a simulation either locally or on remote machine. Also, ODESSI is able to interface with multiple different simulations to conduct more complex investigations such as model-to-model analysis, or analysis across multiple simulations. The simulation manger component employs several workers and a database where each worker controls an instance of the simulation and a database to cache solution for reuse. The simulation manager component act as a server that serves simulation solutions given input parameters.

Scientific methods library. This library contains different algorithms or implementations that implement the scientific investigation methods. For example, several algorithms can be used to implement optimization such as simplex search or genetic algorithms or several procedures can be used in the senstivity analysis.

Planner. Based on the provided scientific method description document, the module generator selects and configures the scientific method and generates a plan to be executed. Planners typically require the response of the simulation for different sets of parameters. Planner gets the simulation response by sending messages to the simulation manger module.

Scripting engine. The scripting engine controls and coordinate the interactions between ODESSI different parts through executing the investigation script.

6.3 Related Work

The general theme of the ODESSI approach is to manage complexity in domain-specific scientific investigations by providing a programmable framework with high-level services for domain contextualization and use.

Problem solving environments (PSE) are a traditional approach to addressing domain-relevant concerns by incorporating all the mathematical, algorithmic, and computational features necessary to solve a targeted class of science or engineering (S/E) problems [73, 162].

The main goal of a PSE is to increase the productivity of scientists by letting them describe a problem and its solution in terms of the S/E concepts and use a highly-functional, integrated set of capabilities for modeling, analysis, and visualization. PSEs have been developed for partial differential equations (PDE) [7], linear algebra [151], chemistry [56], and other S/E areas. However, the traditional PSE approach has three important drawbacks: 1) it is difficult to create a new PSE, 2) PSEs are not developed to be reused, and 3) PSEs are hard to extend with new capabilities or new science methods.

One response to strict PSE design is to identify domain-level functionality that is common across related fields and build software tools that can be applied in developing computational science environments [46]. Scientific development environments take this idea further by offering rich components for data management, analysis, and visualization in a programming framework for scientific applications. For example, SCIRun [193] is a powerful environment for interactive computational science which has been used to create integrated problem solving environments in biomedical science [123]. ODESSI complements these directions by abstracting common simulation-based scientific methods in reusable components, providing a cross-domain framework for scientific investigation.

Grid computing and workflow systems research take a different tact by focusing on how to allocate and coordinate the use of computational resources (both systems and software/tool components) to create and run scientific applications such as GridLab[83]. Grid-enabled workflow systems such as Pegasus[52], Triana [36], and Kepler [122] are powerful tools being applied in computational science projects. However, their support for multi-experiment simulation workflows is still rudimentary and is not easily programmed for cross-domain use or execution on non-grid platforms. Web-based portals (e.g., the NEES [135] and BIRN [24] portals) and environments such as ViroLab [210] address some of these issues by offering higher-level S/E services (e.g., analysis, data management, simulation) while hiding backend complexity. The ability to abstract and reapply scientific methods for new scientific investigations or new scientific domains in these environments though is not supported well.

On the other hand, there are wealth of toolkits for scientific methods used in simulation. The DAKOTA toolkit [157] provides several optimization algorithms, uncertainty quantification, and parameter estimation. The Portable, Extensible Toolkit for Scientific Computation (PETSc) [152] is a suite of data structures and routines for the scalable (parallel) PDE-based scientific applications. The important aspect of these systems is their embodiment of a known scientific methodology in a programmable form. The idea behind ODESSI's approach is to provide a high-level scientific development framework that parameterizes and configures scientific methods for domain specialization.

Our framework design is driven by our domain problems in neuroscience as described in the previous section. There is no single system that provides all the functionality needed to conduct those studies. However there are related environments that provide some of those requirements.

Dakota toolkit [157] is a rich C++ toolkit that provides several optimization algorithms, uncertainty quantification and parameter estimation. It has been demonstrated in several engineering design problems. However, Dakota is focused to provide an engineering design environment for a single program. Dakota is not intended to support distributed workflow in a distributed computing environment. Also, Dakota only considers the model parameters in the analysis and doesnt include the state parameters which are necessary to optimize for the tradeoffs between the solution accuracy and simulation performance. This is important for a class of scientific problems where the simulation execution time is a major factor in solving the problem.

In the grid environments, workflow management systems[76] have an important role in developing applications that utilize the grid available resources to conduct scientific experiments. Several Grid-enabled workflow systems are evolved in the past 10 years. Pegasus [52] is a framework that maps abstract workflow into concrete workflow and it schedules the concrete workflow into distributed resources. Triana [36] is a workflow data analysis distributed environment based on P2P, grid services and web services interaction. Tavera [145]is a service-orieted workflow system in bioinformatics where the components of the application are web services. Condor/DAGMan [67] is a resource management system used to match grid resources to tasks. Kepler [122] is an actor-oriented workflow system. WebFlow [6] and GridFlow[110] and several other PSEs are developed to ease the development of large scale scientific application from a bool of components assembled as a DAG based workflow.

These environments have an important role in building a scientific problem solving environment that utilize the necessary computational resources. However, these environments support only conducting a single scientific experiment or allow a parameter sweep through scheduling. What is missing in these environments is an abstract layer to allow the design of scientific studies composed of several experiments. The user is still required to conduct and manage these studies manually.

Other environments are focused to provide interactivity with the parallel simulation through visualization and steering. SCIRun/UINTAH [193, 226] is a popular bioinformatics problem solving environment that allows rapid interactive design of a scientific experiment. It also provides interactive visualization and steering. CUMULVS [75] is a middleware that allows a programmer to connect remotely to a running simulation, get visualization data and steer a user defined parameters. gViz [99] is a grid enabled visualization tool. In these environments scientist is still required to construct and manage the scientific investigation as a composition of several scientific experiments manually.

In the grid environment, there is little work that supports computational science investigations. Most of this work is limited only to parameter study or parameter sweep by generating several instances of the program corresponding to different parameters and executing these instances concurrently on the Grid or the distributed environment. Nimrod and Cluster [47] are environments that are able to generate and launch parameter study in the grid environment. They are built on top of Globus. They also provide a high level language for describing parameter study creation. ILab [222] from NASA is a graphical tool that supports large parameter study. ILab generates a single shell script for each run in the parameter study. A single directory is created for the whole parameter study and then a sub directory is created for each run where input files are moved to that directory and then the scripts are executed. In case of cluster computing two scripts are generated, the first script remote copies the second script to remote cluster and then it executes there. Similar to Nimrod, AppLeS (Parameter sweep template) [29] is a parameter sweep environment. Its main focus is on scheduling the application on the grid resources in performing the parameter sweep. Similarly Saleve [132] provide a parameter sweep across distributed resources. P-GRADE [112] portal integrates a parameter study with a workflow. It provides a parameter study by considering the workflow as a black box that gets executed with many different parameter sets. In P-GRADE the portal generates the workflow instances and submits them for execution concurrently. Another environment that integrates parameter study with workflow is SciDC[31]. MCell is a grid-enabled parameter sweep application for a biology application [28].

In all these environments the parameter sets are pre-generated and then the response corresponds to these sets is computed. In an on going effort in [212] they extend this by proposing interactive parameter sweep where the user is able to monitor and guide the parameter sets based on intermediate results. However, this approach requires the availability of the user which is not practical in long running simulations.

There is little work that supports other kinds of scientific investigations. Nimrod/O [2, 3] is an optimization framework that supports identifying the engineering design point using several optimization algorithms. Sim-X [223] is an on going effort to provide an interactive optimization tool that allow changing the optimization criteria dynamically and explore parts of the parameter domain while the simulation is executing. SimX is later on added to SCIRun PSE [224]. Again interactive optimization is not suitable for problems that require long execution time.

Most of the research work above is limited to parameter study and in few cases to optimization or standalone application. Our goal is to extend and generalize this work and provide framework support to develop scientific investigations in a way that can draw on standard methods. As discussed below, the ODESSI framework will enable method implementation as programmable modules and their coupling with a simulation planning capability. Parameter sweep, uncertainty quantification, V&V, and comparative methods will be developed. Moreover, ODESSI will provide additional support for investigation scripting and access to database, analysis, and visualization utilities.

CHAPTER VII

ODESSI DEVELOPMENT

Simulation-based scientific investigation in computational science involves several aspects and challenges. These aspects can be broadly classified into three categories, 1) the execution of the simulation on system resources, 2) applying scientific methods and procedures using the response of the simulation, and 3) managing domain data and investigation runs. These aspects generally interact with each other in conducting a scientific investigation in an integrative manner. Typically, a scientific investigation involves selecting one or more domain data sets, apply one or more scientific methods, and require executing the simulation a large number of times with the potential of long execution time. Each one of these aspects has its own complexity and challenges and integrating them in conducting a scientific investigation is a challenge in its own. Increasing the productivity of scientists in conducting scientific investigations can be achieved by raising the level of abstraction in these aspects and raising the level of intergration among them.

Several general purpose systems and tools have been developed to handle a single aspect. Tool kits such as Dakota [157] are focused on applying a scientific method on a simulation, but ignore the domain data aspect. Systems such as Condor [68] are focused on the execution aspect of the scientific investigations primarily. Data management systems and knowledge bases as provided in BIRN [24] are concerned with the domain data aspect. Integrating these aspects in scientific investigation is still, for the most part, done manually. Recently several scientific communities realize the need to integrate these aspects in a single environment which results in developing domain-specific environments such as ViroLab. Although these work well as domain-specific, standalone applications, the environments are not designed to be reused by other domains.

The goal of ODESSI is to provide a generic framework that realizes and integrates the common aspects of scientific investigation that occur in most scientific domains. ODESSI achieves this goal by factoring out the shared concerns of scientific investigations, capturing them as components in the framework, and combining the components as necessary for specific use. Since ODESSI can be extended with domain-specific aspects, a domain computational environment for scientific investigation can be built on top of ODESSI. In this chapter we describe the implementation of the conceptual design of ODESSI framework as described in Chapter VI. From ODESSI's perspective, a computational science investigation is a script that runs and manages all the necessary computation on HPC. It applies one or more scientific investigation method in its lifetime. It defines the simulation codes to use, the input data files, and post-simulation analysis and visualization. We call this script, the *domain investigation script*.

ODESSI is realized on a set of Python objects that implement the components shown in Figure 6.1 in Chapter VI. These objects form a set of interacting threads that cooperate during the execution of the investigation script 7.1. An investigation script instantiates the necessary objects that implement scientific methods that it uses, and these then interact through messages once started. Simulation programs are invoked through the Python system interface. The investigation script consists conceptually of three main sections. In the first section the user specifies the simulations under investigation by providing their names, input/output parameter names and data types, initial values for input parameters, and the specification necessary to execute the simulation including selecting and getting the necessary data files. The simulation specifications are used to create a simulation manager object that can request simulation solutions. In the second section, the scientific investigation methods are customized, instantiated and launched for execution. Each scientific investigation method is executed in a separate thread. Each scientific method object derives from a base Planner object, adding the method specific functionality that it is intended to provide. The third section is concerned with bookkeeping and post-processing the results.

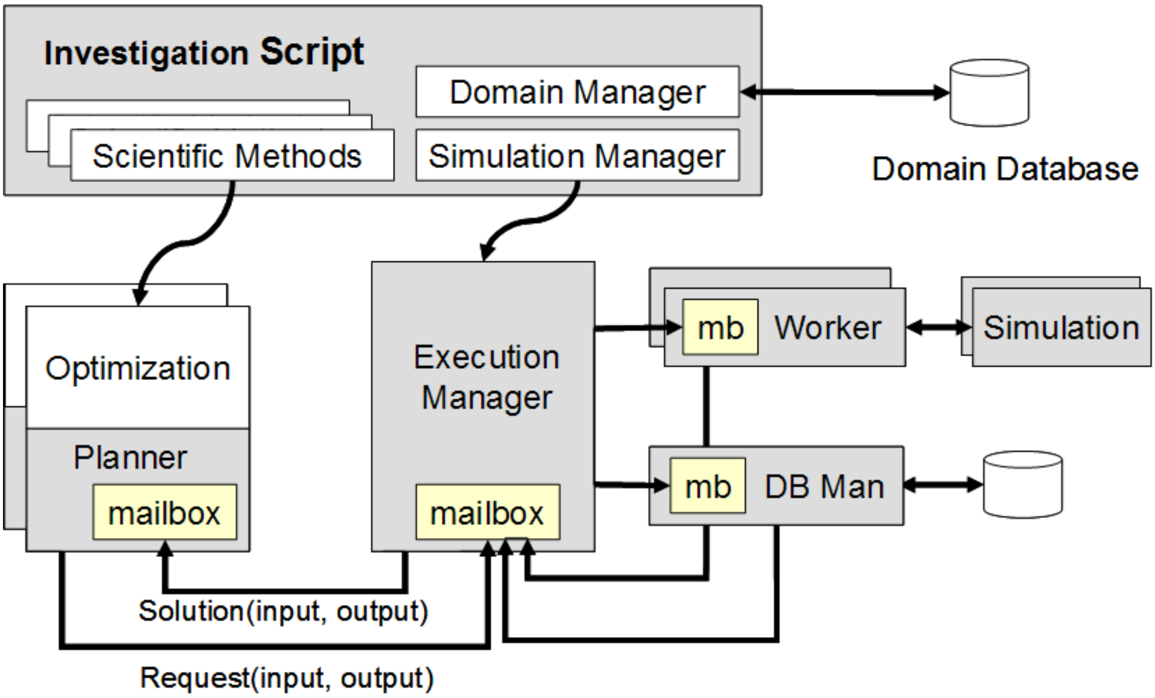


FIGURE 7.1: The investigation script runs the execution manager in a thread and each scientific method in a thread. Scientific methods request solution from the execution manager in there execution life time.

An important part of the ODESSI implementation is the use of asynchronous messaging between threads. Due to the potential for long execution times during simulation runs, it is preferable to use an asynchronous execution model to allow threads to execute independently without causing all others to block. Threads in ODESSI are based on the process model from the Erlang language [10]. We used a Python-based implementation of the Erlang process and message passing model provided by the Python “candygram” package [95]. Erlang is based on the Actor model of concurrent processes that interact via asynchronous message passing. Each process has a message mailbox into which messages from other processes are delivered as shown in Figure (7.1). The receiving process can check its mailbox at any time and extract messages one at a time. Messages are matched with a handler which is invoked to perform some action based on the contents and type of the message. The Erlang programming model guarantees that messages are handled in the order they are received.

Simulation, scientific methods, and data objects are described using the Traits-UI dynamic user interface. Traits and Traits UI are open source Python packages from Enthought[58]. Traits packages provide an easy way to give types to variables with bounds checking. This is helpful to catch bugs in scientific computing. Also, Traits support reactive programming model which is oriented around data flows and propagation of change. This model is useful in writing scientific applications, in which arrival of new data triggers change on other variables. This model can be used in implementing real-time monitoring of the scientific method execution. Traits-UI provides a graphical representation of an object which allows automatic generation of dialogs to edit the attributes of Python objects. It provides a rich list of editors and the mechanism to build interactive user interface. This structure maps well to scientific applications in general and to ODESSI environment in particular where the specification of scientific methods, domain data, and the simulation are represented as objects. These objects can be viewed and edited directly with Traits UI editors. The graphical representations view of the object in Traits-UI reflects the structure of the objects which makes it easier to extend.

Since one goal of ODESSI environment is to provide rapid prototyping environment for simulation-based scientific investigation, an interactive Graphical User Interface (GUIs) is desired. Creating such a user interface through a toolkit doesn't integrate well with general purpose environment such as ODESSI since the objects are likely to be changed and extended frequently. Further, creating first a graphical interface and then writes the callbacks of the graphical objects cause messy mixed up between the model, control and the view of the model and limits the extensibility of the environment. The Traits-UI method of interface construction has been helpful in addressing these issues.

ODESSI provides three entities that can be instantiated in the investigation script: the *Simulation Manger* entity, the *Scientific Investigation Method* entity, and the *Domain Manager* entity. The Simulation Manger is the entity that controls and manages the execution of a simulation. It acts as a server that provides solutions given sets of input parameters. Each instance of a simulation manger controls a single simulation. Multiple simulations can be controlled by multiple instances of the simulation manager. A simulation manager object can serve multiple requests from different threads. Scientific investigation methods provide the scientific investigation procedures that are common in several scientific domains such as optimization and sensitivity analysis. When a scientific investigation method is instantiated, it gets executed in a thread that interacts with other threads. The Domain Manager is the entity that manages a scientific domain. This includes getting and setting domain data objects, and domain simulation objects. In the following 3 sections we describe in details these entities.

7.1 Scientific Investigation Methods

Computational simulation has become the preferred methodology for conducting scientific investigations in many domains (e.g., aircraft design, electronics, chemistry, physics). The common requirement to these domains is the availability of affordable computing power in terms of hardware, computational models, and scientific investigation methods. Scientific investigation in computational science involves the application of several methods such as optimization and sensitivity analysis. These methods are used to explore the model parameter space, to rank the parameters according to their significance, to extract optimal model parameters, and to verify and validate the results. Typically, there exist several algorithms for each scientific methods (e.g., several optimization algorithms) that can be applied, and generally the quality and performance of these algorithms are highly dependent on the model under investigation. Which means an algorithm may perform well on a particular problem, but poorly on a different problem, and vice versa. Further, for each algorithm, the user must specify a set of parameters empirically which usually involves a tradeoff between the quality and the performance of the algorithm. Also, as science evolves, new algorithms and new scientific methods are always developed. Due to the large number of options, scientist are faced with the problem of choosing the best algorithm and the optimal configuration parameters that provide the optimal performance and the highest quality of investigation for their model.

The conventional approach of scientific investigation is by developing standalone application using software libraries for each investigation. While this approach achieves the desired goal, it has several limitations. First, it is time consuming and limits the productivity of the user since several applications must be developed and maintained for several investigations. Second, the user must reengineer the application when the underlying model is changed, the investigation methods are changed, the method algorithm is changed, or when experimenting with new algorithm. These changes are common and frequent in a rapidly evolving field like computational science. Third, the user must develop tools or customize (which can be hard in general) existed tools to utilize the distributed computing power in executing the simulation because the execution time of most scientific application is long. These factors limits the productivity of scientist and more important it limits the quality of science, since scientists might get satisfied with any satisfactory results without experimenting other methods or algorithms which might provide better quality or performance. These limitations argue for better approach in conducting scientific investigation.

Our approach to scientific investigation is to factor out the common scientific methods in a generic framework where they can be applied on arbitrary simulation and reused by different domains. This approach is extensible where new methods can be added independent of the simulation. Further, the specification and the execution of the scientific method are separated, this separation allow rapid customization of the method using other tools such as Traits-UI, and also, allow saving the specification with the result which provide provenance information. The separation of the method execution in its own thread allow multiple methods to be executed concurrently while sharing simulation execution which results in better utilization of system resources. This approach allows a scientist to experiment with different methods and configuration parameters to investigate the underlying scientific model rapidly. This will increase the productivity of scientist and results in improving the quality of science.

In this section, we describe how the scientific methods are implemented, customized and executed as part of a scientific investigation. We demonstrates our approach using three common scientific methods, 1) optimization based on a parallel simulated annealing algorithm, 2) parameter sweep and 3) linear regression based sensitivity analysis. ODESSI currently supports these methods and can easily be extended with more methods and procedures. In ODESSI, a scientific method gets executed in its own thread spawned by the main thread in the investigation script. The scientific method is implemented as a module of three classes: *Specification class*, *Interface class*, and *Logic class*. These classes are described below.

Specification class. A specification class provides a template for the user to customize the scientific method. The specification class gets instantiated, customized and passed to the scientific method interface class in the main thread. The specification class object can be specified either through the constructor in the investigation script or using a Traits-UI [201] dynamic GUI editor or both. When the *custom* argument of the specification class constructor is True, a Traits-UI user interface pop up and allows the user to customize the parameters. For example, Figures 7.27.3 shows a demo of Traits-UI pop up to customize a parametric study and optimization methods. When the GUI interface pop up all parameters of the simulation inputs appear with their initial values. Then the user can customize these parameters through the UI interface. The investigation script blocks until the user finishes customizing the method.

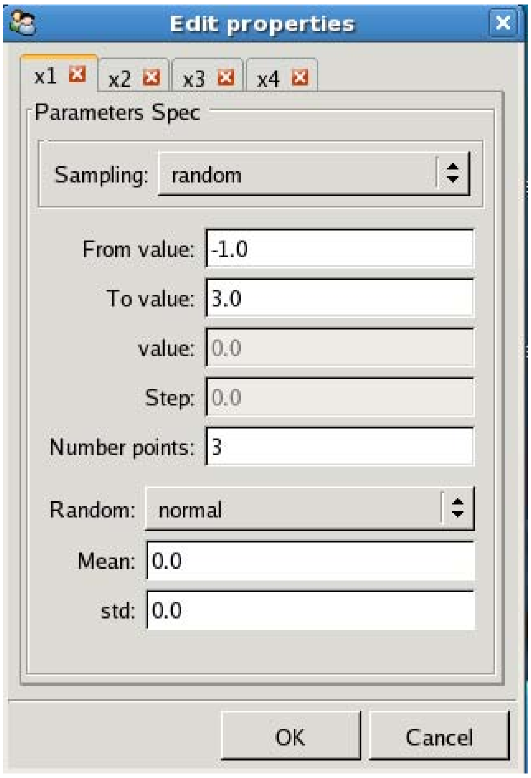
Interface class. An interface class provides an interface for the user to instantiate and executes the scientific method in an independent thread. An interface class takes an instance of the method specification class and a simulation manager object. Its purpose is to manage the interaction between the investigation script and the scientific method which is executed in a separate thread. The investigation script interacts with a running scientific method by calling methods provided by this class. Currently the interface class provides three methods, *go*, *stop*, and *get_results.* The go and stop methods starts and stops the execution of the scientific method. A call to get_result blocks until the execution of the scientific method completes and the results are returned to the investigation script. These methods are inherited from a MethodInterface base class. Further functionality of this class can be added such as monitoring the execution of the scientific method and visulizing intermediate results.

Scientific method logic class. The scientific method logic class implements the algorithm or the procedure of the scientific method. It must have a Planner class as a base class. This class gets instantiated and executed in its own thread by the scientific method interface class. The scientific method specification and the execution manager object are obtained in the initialization. The scientific method uses two methods inherited from the Planner base class to get solutions of the simulation, *request_solution* and *get_solution*. The request_solution method has three arguments, a tag to identify the returned solution, a dictionary of input parameter names and their values, and a list of the output parameter names. Not all the inputs of the simulation need to be specified. Any unspecified input will take its initial value as specified in the investigation script. The get_solution method is used to get one returned solution for each call. It takes no arguments and returns a list of three items, a tag corresponding to the request tag, a dictionary of the input parameters and their values and a dictionary of the output parameters and their values. If there is no solutions returned yet, the get_solution method blocks until a solution is arrived. All scientific method logical classes must have an abstract base class called Planner. The planner class provides the interactions between the scientific method and the execution manager. An instance of the execution manager that controls a simulation is passed to the Planner class in the instantiation of the scientific method. The Planner base class acts as a stub for the scientific method logic class. Therefore, the scientific method logic code is completely separated from the communication code and is not of the user concern who desire to add new scientific method. Once simulation are initiated, the Planner object does not guarantee the order of return of solutions. This is our design decision as many of the scientific methods are Monte Carlo based and the order is not important. However, the Planner class includes an optional feature to enforce simulation dependencies if necessary. When the the Planner *request_solution* method called, it delivers the parameters as a message in the execution manager mailbox. When the *get_solution* method called, it extract a message from the its mailbox and return the message to the caller.

In the following subsections we describe the scientific investigation methods currently supported by ODESSI. One long-term objective of ODESSI is to provide a rich list of scientific methods that are common in most scientific domain.

7.1.1 Parametric Study Methods

Parametric studies are used to explore the parameter space for better understanding the model behavior under different conditions before or during more complex analysis such as verification and validation or sensitivity analysis. They are pursued to explore the effect of one or more parametric change on the simulation



Parameters Spec

Sampling: random

From value: -1.0

To value: 3.0

value: 0.0

Step: 0.0

Number points: 3

Random: normal

Mean: 0.0

std: 0.0

FIGURE 7.2: Parametric study specification

output to study cause-and-effect relationships. It also underlies Monte Carlo simulations and can be used in computing derivatives and the simulation outputs required by other analysis or applications. Typically, a parametric study is a necessary step before any investigation. In parametric studies, one application is executed many times with different sets of input parameters. These runs are independent and can be executed concurrently. ODESSI simplifies this kind of analysis by automating the time consuming, manual execution of the simulation on distributed computing resources and gathering the results efficiently in an easy to use structure were further analysis can be applied. ODESSI can utilize any number of distributed resources in performing this computation without any requirement on the user part. Currently, ODESSI provide several parameter space sampling methods. The list of choices can easily be extended.

Line space. In line space sampling of a parameter, the user specifies the *start/end* values and the number of sampling points n. Then ODESSI generates the n-sample points with uniform interval between them, and computes the simulation response at these points. The following is an example of linespace specification,

```
Param('x', fromvalue=10, tovalue=20, npoints=10,
      sampling='linspace')
```

Log space sampling. Log space sampling is used to sample a parameter on a log scale instead of a linear scale. Log space sampling is similar to line space sampling except the *start/end* values are the exponents of the *base* (e.g. from $base^{start}$ to $base^{end}$). Log space sampling is useful when exploring parameters on a wide range (e.g. from .001 *to* 1000000),

```
Param('x',fromvalue=1, tovalue=3, npoints=10, base=10,
      sampling='logspace')
```

Range. In range sampling the user specifies the *start/end* values of a parameter and the uniform distance between the points (*step*). Then ODESSI generates the necessary number of points that cover the range,

```
Param('x', fromvalue=1, tovalue=3, step=2, sampling='range')
```

Random. In random sampling, the user specifies the *start/end* values of a parameter, the number of sample points, and the sampling distribution with the corresponding distribution parameters. For example, in case of random normal sampling the user specifies the mean and the standard deviation. ODESSI currently support uniform and normal sampling. Other random distributions can be added in a straightforward manner. Random sampling is necessery in senstivity and uncertainty analysis.

If more than one parameter is specified in a parametric study as described above, then a multi-dimensional cube is formed. Each axis of the cube corresponds to a parameter. Then ODESSI computes the response of the simulation at every point in the cube. For example, if parameter x is specified to take 10 linespaced values and parameter y is specified as a 'range' which results in 5 values, then a square is formed, and the simulation response is computed at all $5 \times 10 = 50$ points. If n parameters are specified with each parameter takes m values, the results is computing the simulation m^n times which can be very large. Therefore, distributed computed is required to conduct this kind of analysis even for simulation with relatively short execution time (less than a minute).

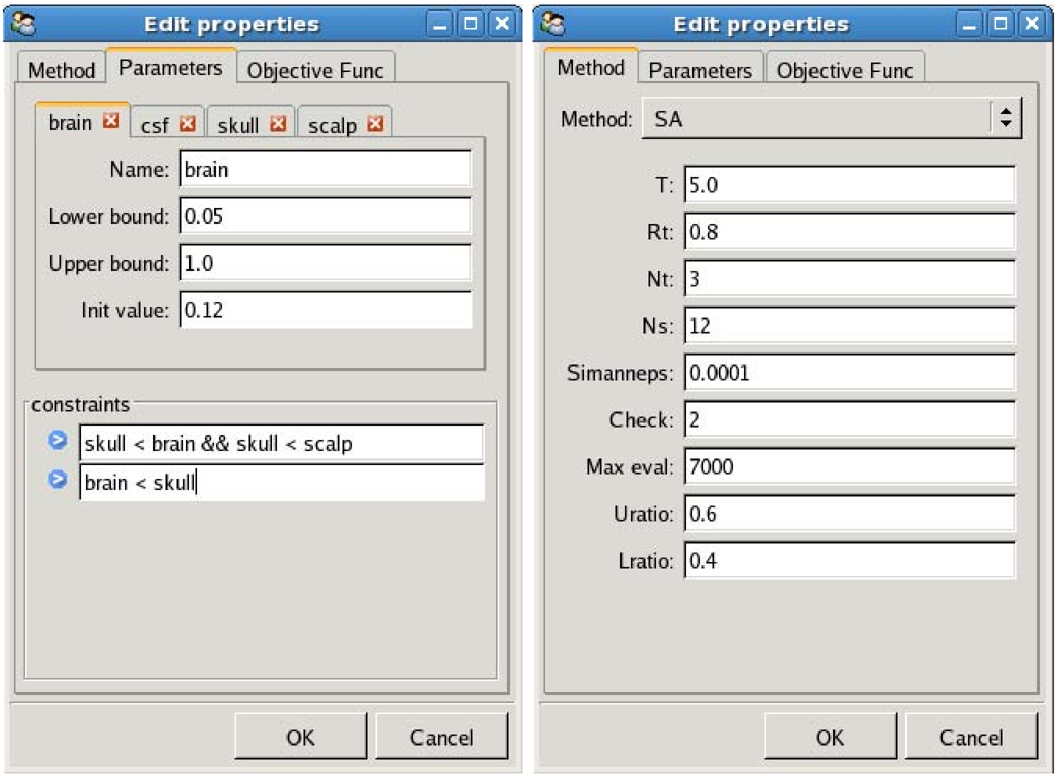
ODESSI uses the default values of the parameters as specified in the simulation specification for any unspecified or partially specified parameters. Unspecified parameters take their default values and unspecified start/end values take the lower/upper bounds as specified in the simulation specification. If the mean of a normal random sampling is not specified odessi uses the center value (the value at the middle of the start and end values). If the standard deviation is not specified odessi uses the average distance between the mean and the lower and upper bound divided by three.

These sampling options can easily be extended to support other sampling strategies, such as, to perform parametric study along arbitrary line in the parameter space, or around a pre-selected point, or at points selected from a user specified list. In addition to the ability to specify these parameters in the investigation script as described above, the user can invoke the parametric study module with the optional argument ($custom = true$). This causes a GUI interface to popup which displays all simulation input and output parameters with their default values, and their lower and upper bound. Using the GUI interface, the user can customize these parameters easily without reading any user manuals. Figures 7.2 shows a GUI user interface popup when running a parametric study on two different simulations. Chapter VIII provides more concrete examples.

7.1.2 Optimization

Global optimization is the problem to find the globally best optimal solution of a model in the presence of multiple local optima. The optimization problem can be stated as follows: Find the values of the best optimal parameters that optimizes (minimizes or maximizes) a multi-dimensional objective function and subject to some constraints. The maximization of a problem can be treated as the minimization of its negative. The optimization problem takes the form: $Optimize(F(\mathbf{x}))$ *subject to* $\mathbf{l} < \mathbf{x} < \mathbf{u}$, where $\mathbf{x}$ is the model N-dimensional variables vector. These variables are independent and a change in a variable cause a change in the model response. F is a cost or objective function that depends on the model output. $\mathbf{l}$ and $\mathbf{u}$ are the lower and upper bound vectors. The N variable input parameters define the optimization search space. An optimization algorithm searches the model input parameter space for the set of parameters that optimizes the objective function. In general the objective function of realistic problems is non-linear, and the search space contains many local minima's and possibly multiple global minima.

Optimization problems occur in every scientific domain. One important class of optimization problems are inverse problems. The aim of an inverse problem is to determine the values of the underlying model input parameters that explain a given observed data. Therefore, in these problems the objective function is the difference between the observed data and the model output. The complexity in these problems is due to the noise in the measured data and the numerical noise in the model output (e.g. due to discretization). This results in unclear global minimum and so the optimization problem must to be solved several times with different starting point or different configuration of the optimization method. Typically, in these



Name: brain
Lower bound: 0.05
Upper bound: 1.0
Init value: 0.12

skull < brain && skull < scalp
brain < skull

Method: SA
T: 5.0
Rt: 0.8
Nt: 3
Ns: 12
Simanneps: 0.0001
Check: 2
Max eval: 7000
Uratio: 0.6
Lratio: 0.4

FIGURE 7.3: Optimization specification interface: Optimization problem specifications and setting the objective function (left). Selecting and customizing the optimization method (right).

kinds of problems and in most science and engineering optimization problems, the evaluation of the objective function is the costly step and the performance is totally dominates by this step.

Several optimization algorithms are developed over the past few decades. However, the performance of an optimization algorithm is problem-dependent and varies from one problem to another. Actually, it is believed that there is no optimization algorithm exist that can perform well on every optimization problem. One algorithm might perform the best on a particular problem, but the worst on a different problem. Therefore, choosing the best optimization algorithm for a particular problem is a problem in its own. Further, each optimization algorithm requires setting several configuration parameters to tune the performance and the quality of the algorithm. Typically, setting these configuration parameters involve a tradeoff between the quality and the performance of the algorithm. One goal of ODESSI design is to allow experimenting with different algorithms and with different configuration parameters for each method rapidly without developing new application for each case. This allow users to make a better choice of the best method with the best configuration parameters for their particular problem.

ODESSI allows the user to specify the optimization problem, the optimization method, and to customize the optimization method either through the investigation script or using the dynamics Trait-UI editor as shown in Figure 7.3. In the “Parameter” tab in Figure 7.3, the user can specify which parameters to include in the optimization, their lower and upper bounds and their initial values. Also, the user can impose constraints on the parameters which allow the optimization algorithm to avoid searching the regions that violate the constraints. In the “Method” tab the user can select the optimization method to use and configure the method parameters. The figure shows the configuration parameters of the simulated annealing algorithm as described below. In the objective function tab, the user can specify the Python code for the objective function.

Using only this simple interface, the user can experiment with a variety of different scenarios including, different initial values, lower/upper bounds, constraints, objective functions, optimization algorithms, optimization algorithm configurations. Also, the user can experiment with different optimization parameters and fix other parameters at specific values. ODESSI will take care of all the necessary computations on distributed resources without any requirement on the user side. Same experimentation can be performed on other models without writing or changing any code.

Currently, ODESSI support optimization with simulated annealing algorithm. The parallel simulated annealing algorithm implemented in ODESSI, is described in details in Chapter V. In the following we provide a short review with the focus on how to set the algorithm configuration parameters.

Simulated Annealing Algorithm

Simulated annealing (SA) algorithm is based on a Monte Carlo simulation that simulates the physical process of annealing. It can be considered as a modification to the hill climbing algorithm by adding a stochastic decision and a cooling schedule. Similar to hill climbing algorithm, the trial solution is always accepted if it performs better, but inferior solutions can be accepted with some probability as well. This stochastic decision allows the algorithm to escape from the local optima. The acceptance probability is a function of the system temperature and it decreases over time, $P = exp(-F/T)$, where F is the increase in the cost function and T is the temperature in the simulation (a parameter that controls the acceptance probability of the worse solution). Initially when T is high the algorithm accepts inferior moves with higher probability. However, as the system cools down, moves to inferior solution are accepted with smaller probability. The simulated annealing algorithm consists of three nested loops

Temperature cooling: The outer loop controls the temperature cooling; the temperature is reduced by a factor of r after executing Nt search-radius loops. Other cooling schedules are proposed in the literature.

Neighborhood Search radius: In a continuous problem the neighborhood of a solution are those solutions that can be reached within a certain radius in the search space. The size of the search radius determines the characteristics of the search. Large radius causes the search to be explorative and less likely to converge to an accurate solution. On the other hand, small radius causes the search to be exploitive and less likely to explore the search space well. Therefore, normally an adaptive mechanism is used to adjust the search radius such that early in the search, it is large (explorative) and it decreases as the search gets closer to the solution (exploitive). In our implementation, the search radius is adjusted based on the rate of the accepted to the rejected moves as propose in [39]. In this approach, when the number of accepted moves increases the search radius increases and when the number of rejected moves increases the search radius decrease.

Control point: The inner loop considers new moves in all directions by perturbing the control point in all direction. The perturbation is constraint between 0 and the maximum step length in each direction. Transitions that lower the cost function are always accepted, while transitions that raise the cost function are accepted with probability based on the temperature and the size (cost) of the move (Metropolis criteria). The simulated annealing converges when the minimum become stable that does not change for more than epsilon after several temperature reduction loop iterations. The complete algorithm is given in Algorithm1. The choice of a suitable initial temperature, search radius and the cooling schedule are highly problem dependent and normally are chosen empirically and they have a significant impact on the performance of the algorithm. Below is a short summery on how to guide the parameters (shown in Figure 7.3) selection [172].

- maxiter: This is the maximum number of model evaluation. The algorithm returns the best optimal value obtained so far when the number of model evaluations reaches this number.
- simanneps: This is the convergence criteria of simulated annealing. Simulated annealing converge if the change in the accepted solution remains less simanneps for a “check” temperature reductions iterations.

- T: This is the initial value of the temperature. It is used to control how fast the simulated annealing search proceed. High T values, results in acceptance of most uphill moves (like random search-exploration of the search space). When initial temperature is low, uphill moves are mostly rejected.

- r_t: This is the temperature scheduling; the temperature is reduced by this factor at the end of the temperature loop. Large r_t result in reducing the T slowly, which means the search proceed slowly and explores the search space better. Lower r_t value means reducing the temperature faster which results in faster converegence but maybe to a local minimum. r_t must be between $0 - 1$.

- N_t: The number of loops before temperature reduction. High values results in better exploring the current search area, but more computations.

- n_s: Number of loops before adjusting the search radius. The search redius is adjusted such that half of the moves is accepted.

- vm: The initial maximum step length. vm is adjusted at the end of the radius search loop. When simulated annealing select a trial point the point will be within the search radius.

- cstep,lratio and uratio: These parameters are used in adjusting the maximum step length (search radius) at the end of the ns loop. The algorithm adjust the maximum stepp value for each direction based on the number of accepted moves and number of rejected moves 5.5. For instance, to keep the ratio between the accepted to rejected moves between .4 and .6, we set the laratio to .4 and uration to .6. §5.5 for more details.

- Check: The number of temperature loops to check for converengence. If the change in the optimal point is less than simanneps then the algorithm terminates.

ODESSI allows running multiple scientific methods or multiple instances of the same method concurrently assuming the availability of sufficient computational power. In ODESSI concurrency can be achieved in two ways, 1) by applying the scientific method multiple times with different parameters and/or different initial values and 2) by parallelizing the scientific methods itself. Since ODESSI provide the mechanism to utilize any amount of computing resources, maximum performance can be achieved by applying both types of concurrency and so it is desired to parallelize the scientific methods to increase the degree of parallelism. Once the method is customized and launched for execution, the method requests model solution from the execution manager whenever it needs a solution. ODESSI uses a parallel version of the simulated annealing as described in section 5.5.2.

7.1.3 Sensitivity Analysis

A modern scientific simulation typically has many input parameters, but in practice the simulation response is dominated by only a few of them. The response can be highly sensitive to small changes in some of them and insensitive to changes in others. The study of how the variation in the output of a model can be apportioned to different sources of uncertainty in the input parameters is the subject of the sensitivity analysis [183, 182]. The goal of the analysis is to quantify the sensitivity of the model output due to uncertainties in the input parameters or model assumptions.

The outcome of a sensitivity analysis is a ranking of the model input parameters according to their significance/contribution to the model output. This information is valuable for multiple purposes: 1) improving model robustness by identifying the critical regions of the parameters space and uncovering errors in the model, 2) prioritizing the most influential and relevant parameters to focus research on improving the uncertainties, 3) simplifying the model by reducing the number of parameters which allow more efficient computation for other investigation (e.g. optimization) by fixing the values of the insensitive parameters, 4) clarify the interaction between the input parameters which allow better understanding of the model behavior, and 5) rule out the investigator bias. In general, sensitivity analysis is useful for any process that needs to know the most influential parameters on the model output variability such as in verification and validation processes. A formal definition of global sensitivity analysis that captures these goals.

> Senstivity analysis (SA) is the study of how the variation in the output of a model (numerical or otherwise) can be apportioned, qualitatively or quantitatively, to different sources of variation[183].

In the literature, one finds several sensitivity analysis techniques. The earliest and most intuitive techniques are local methods. Local methods provide estimation of the effect of local variations in an input parameter on the model output. Thus, their main concern is with input parameters that are known with certainty. Local methods are based on the partial derivative computations of the model output with respect to a model input parameter. This derivative is used as a sensitivity measure. These methods provide informatics results for linear models and only when the input uncertainties are locally restricted to a narrow region. They suffer strong limitations when the models simulate nonlinear phenomena and when the response cover a wide input parameters range, which is the typical case for science and engineering phenomena.

Recently, most sensitivity studies are focused on global methods. These methods analysis the entire range of variation of uncertain input parameters (they analyse the whole parameter space). They are based on random-sampling Monte Carlo techniques which allow the computation of each parameter's contribution to the model output variance. Therefore, our main focus is on the global sensitivity analysis approaches. Global sensitivity methods are typically classified into two categories, regression-based methods, and variance-based methods.

1. *Regression-based methods:* Correlation coefficients measure the linear effect of changes in the input parameters on the model response. Therefore, they can be used as a measure of global sensitivity. The Standardized Regression

Coefficients (SRC) are exactly the correlation coefficient between the dependent and the independent variables. They are based on a linear regression of the output on the input parameters. However, in case of general nonlinear and nonmontonic models, the sensitivity ranking results of these approaches can be misleading. Therefore, these approaches are useful when the model has strong linearity.

2. *Variance-based methods:* In these methods, the variance of the model output is decomposed into factors induced by the input parameters. This decomposition is usually referred by *ANalysis Of VAriance* (ANOVA). The most common decomposition is Sobol' decomposition which produces global sensitivity indices. Sobol's indices provide satisfactory results for the general nonlinear and nonmonotonic models.

In this dissertation we focus only on the Standardized Regression Coefficients (SRC) global sensitivity analysis which ODESSI currently support, other approaches are to be added.

7.1.4 Standardized Regression Coefficient (SRC)

The Standardized Regression Coefficients (SRC) method is based on multiple regression analysis of the model output on the input parameters. Each SRC coefficient quantify the change in the model output per unit change in an input variable when all other variables are held fixed. It provide a measure of the strength of the linear dependence between the dependent variable and an independent variable. The SRC method provides a good approximation of the global sensitivity measure when the underlying model possess strong linearity. One advantage of the method, is its relatively low computational cost compared to other global methods since their performance is independent of the number of input parameters. The SRC method consists of three steps: 1) a multivariate samples of size n of input parameters, $X_i, i = 1, \cdots, k$, is generated using some sampling method, 2) the model output, Y, is computed by running the simulation with inputs given by the samples, 3) the output vector Y and each input parameter vector (X_i) are standardized, so that they have a variance of one and a mean of zero, by subtracting the mean and dividing by the standard deviation, $\hat{Y} = (Y - \bar{Y})/s_y$ and $\hat{X}_i = (X_i - \bar{X}_i)/s_i$, where the hat denote a standardized variable, and 4) a multiple linear regression of the standardized output, $\hat{Y}$, on the standardized inputs $\hat{X}$ is applied,

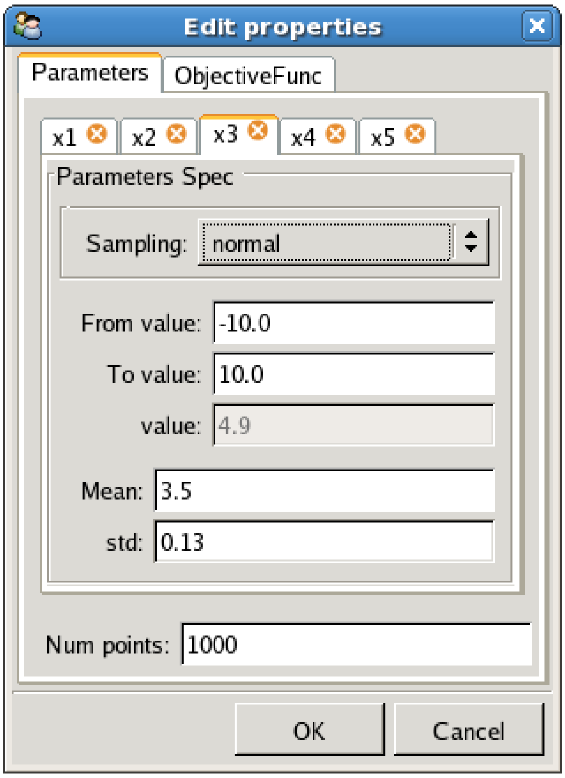
$$\hat{Y}_i = \beta_0 + \sum_j \beta_j \hat{x}_{ij} + \epsilon_i,$$

where $\beta_j, j = 1, \cdots, k$ (k being the number of input variables) are the standardized regression coefficients to be determined, and ϵ_i is the residuals. The least square method is one common way to determine the coefficients β_j by minimizing the sum of the square of residuals, $\sum_i \epsilon_i$. Once the β_j are determined, they provide a measure of the influence of an individual input variables X_j on the variance of the output Y. Since each standardized regression coefficient represents a change in the model output per standard deviation change in an input variable, the coefficients are the same regardless of the independent variable units or scale. Some statistical packages such as R and Matlab provide β_j automatically in addition to the ordinary unstandardized coefficient b_j (the coefficient of regression performed on on unstandardized variables). β_j and b_j coefficients are related by the formula, $\beta_j = (s_{x_j}/s_y)b_j$, where s_y and s_{x_j} are the standard deviations of the dependent and independent variables. Since β_j are multidimensionality averaged measures, they are considered global measures. However, to explore the parameters space a large number of samples must be used.

When using the SRC method it is important to consider the level of the model linearity. The fraction of the model linearity is given by the *model coefficient of determination (R_y^2)* which is equal to $\sum(\beta_i)^2$,

$$R^2 = \frac{\sum(\tilde{y}_i - \overline{y})^2}{\sum(y_i - \overline{y})^2},$$

where $\tilde{y}_i$ denotes the estimate of y_i obtained from the regression model. R_y^2 represents the fraction of the variance of the model original data explained by the regression. The validity of the SRC measure is conditional on the value of R_y^2. The closer R_y^2 to one, the better is the results. For linear models, R_y^2 is equal to one, but for nonlinear models, it is less than one. In practice if R_y^2 is less than .7, SRC as a



Sampling: normal

From value: -10.0

To value: 10.0

value: 4.9

Mean: 3.5

std: 0.13

Num points: 1000

FIGURE 7.4: Sensitivity analysis specification interface

measure of sensitivity is considered invalid. For example, if $R_y^2 = .8$, then the model is 80% linear which mean 80% of the variance in the output is explained by the regression model, but the rest 20% are ignored.

7.1.5 Sensitivity Analysis Using ODESSI

ODESSI currently supports the SRC-based sensitivity analysis. Since sensitivity analysis is defined for a single valued output, the user first defines an objective function that depends on the simulation output. Figure 7.4 shows the user interface to customize the sensitivity analysis method. Under the "Parameters" tab the use can selects the parameters under investigation from all available model input parameters. For each parameter, the user specifies the upper and lower bounds, the distribution to be used in sampling the parameter and the corresponding parameters associated with the distribution. Also, in the "Parameters" tab, the user can specify the number of samples to be used in the analysis. Typically, at least 1000 samples are necessary to conduct a useful analysis. Larger number of samples will cover the space better, and produce more accurate results. Currently ODESSI supports only uniform and normal random sampling. Other sampling methods can be added as discussed in §7.1.1

Once the objective function and the model input parameters are specified. ODESSI generates the matrix as discussed in section §7.1.4. Each row corresponds to computing the model at the inputs. Each column corresponds to sampling a parameter from the specified distribution. The first n-column in the matrix corresponds to the n-input parameters, and the rest corresponds to the objective values. After computing the matrix using the execution manager on distributed resources, ODESSI interfaces with R package and computes the multidimensional regression fit for each objective value and returns to ODESSI the parameters with their corresponding senstivity coefficient corresponding to each output.

Using this simple interface only, the user can conduct sensitivity analysis and study the sensitivity of the model output to different input parameters under different scenarios. All distributed computing required to compute the matrix and the interfaces required to use other packages such matlab or R are completely abstracted out. Further, once we add other sensitivity analysis methods, the user will be able to experiment and compare the results obtained using different methods. ODESSI allow conducting this analysis concurrently with other analyses such as optimization or other instance of the sensitivity analysis but with different setting such as different sampling distribution, or different lower and upper bounds, or different mean and standard deviation.

7.2 Execution Model

Scientific simulations use high resolution computational models to achieve accurate modeling of real world problems. This results in long execution time which typically ranges from minutes to hours. Simulation-based scientific investigation requires solving the simulation many times (thousands of times) for each investigation. Each investigation is usually repeated multiple times to experiments with different parameters. As a result, tens or hundreds of thousands executions of the simulation are required to conduct several investigations to provide answers to scientific questions. However, most of these computations are independent and can be done concurrently on distributed resources. Concurrency in these computations can be achieved in two ways, 1) by concurrent execution of multiple scientific methods and multiple instances of the same scientific method (with different parameters) , and 2) by parallelizing the scientific investigation method itself.

Therefore, any computational environment for scientific investigations must support distributed computing. As mentioned before, and shown in Figure 7.1, ODESSI supports distributed computing by explicit separation between the execution of the simulation and the scientific investigation methods and then couple them together through the Simulation Manager entity. This design has several advantages:

- **Abstraction of computation.** Developers of new scientific methods do not need to worry about distributed computing, all they need to do is to invoke a method provided by the base class of the scientific method which allows them to focus on the logic of the method rather than on the execution aspect.
- **Separation of concerns.** Modification to the execution model does not require any modification to the scientific methods and modification to the method does not require modification to the execution or data models.
- **Better utilization of system resources.** All scientific methods in an invitigation share the use of resources, and so synchronization in a parallel methods will not cause resources to be idle
- **Robustness.** Failure of a node can be handled by the execution layer instead of failure of the scientfic method.

- **Scalability.** When new resources are added to the environment (purchasing new computer), nothing need to be modified to use them, all is needed is to register the new resources with ODESSI.

- **Interoperability.** The architecture and operating system of the resources are not of concern to ODESSI, once the simulation is compiled and can run on the new a resource, the resource can be used by ODESSI, whether the resource is a PC, MAC, linux box, cluster, GPGPU device, or cell-broadband engine. All these heterogeneous resources can be utilized in conducting scientific investigations.

- **Reuse and sharing of previous solution.** The execution layer can employ a database to store previous simultion executions and reuse these solution upon request by any method.

In this section we describe how to specify and describe a simulation object. Then we describe the implementation of the simulation manager object that is used to manage and control the execution of the simulation.

7.2.1 Simulation Specification

A simulation manager object controls the execution of a single simulation. A simulation object is created by specifying the simulation characteristics such as the simulation name, the simulation input/output parameter names, their data types and their initial values. In addition to simulation characteristics, the execution characteristics are specified. These include information about the resources for execution such as the execution site address, the simulation binary path on each site, and the working directory on each resource. The simulation object is defined in terms of its input and output. Conceptually the simulation inputs can be classified into three kinds: data file inputs corresponding to data files, constant parameter inputs corresponding to complex data types such as arrays or tables, and variable inputs corresponding to atomic data types. Data file and constant parameter inputs can not be varied during the execution of a scientific investigation method since investigation methods are defined for atomic data types such floats or doubles. The variable inputs are the independent parameters that can be varied in conducting controlled numerical experiments, for instance, to perform sensitivity analysis or optimization.

Data files are always passed to the simulation by moving the data file to the executions site working directory and the name of the file is passed as a constant parameter. Data structures such as arrays or tables are passed to the simulation as text. ODESSI provide default generic translator functions that translate multidimensional arrays and tables structures to text which can be written to a pipe. The default translator for multidimensional arrays translates array object to a string assuming row order C style arrays. The table structures default translator writes the rows of the tables separated by the end line character. ODESSI also allow users to define their own translators in case the simulation expects different format, for instance, further header information, column order arrays instead of row order arrays, or other user defined structures that are not defined in ODESSI. Before ODESSI passes the input parameters to the simulation, it checks if a user defined translator is provided, if so it uses it, otherwise it uses the default translator. Default translators for atomic data types are the trivial ones, which convert the data type to a string with the specified precision in case of float and doubles. The following script code snippet demonstrates the design in a more clear way.

```
 sim_name = ‘‘forward’’
```

This line specifies the binary executable name of the simulation, This line specifies the binary executable name of the simulation,

```
lsim_inputs_types = {’geom’:(’datafile’, str), ’tol’:(’atom’, float),
                     ’maxiter’:(’atom’ int), ’brain’:(’atom’, float),
                     ’skull’:(’atom’, float), ’scalp’:(’atom’, float)}
```

This line defines the input parameters data types, the ’geom’ parameter is defined as a data file, so the ’geom’ input parameter expects a file name. ODESSI handle this type by moving the file to the working directory and passes the file name as parameter to the simulation. The rest of the parameters are atomic parameters with their specified types. These atomic parameters are to be used in the investigations to conduct, for example, sensitivity analysis or optimization or any other scientific investigation method. In this definition we recognize that 'tol' and 'maxiter' are independent variables associated with the simulation itself, they are not associated with any other variables. On the other hand, the variables, 'skull', 'scalp' and 'brain', are atomic variable associated with the geometry data file (the 'geom' parameters) in which different geometry can have different set of associated parameters, for instance, we might have a geometry that has four tissues (brain, csf, skull and scalp). In this case we need to redefine the simulation and add the fourth parameter. This solution is acceptable when we consider conducting a rapid investigation on a simulation for a specific study. But, it becomes inconvenient when building a domain investigation environment which requires defining the simulation only once. The solution to this problem as described in section [] in the context of building a domain investigation environment is by defining a data object type. The data object type contains a list of parameters associated with the data object in addition to other information (e.g. metadata). Once the data object is selected the simulation input parameters list is extended with the data object associated parameters. In this case the simulation is needed to be defined only once.

```
sim_output_types = {'pots':('table', [('id', int), ('pot', float)],
                    'numiter':('atom', int)}
```

This line defines two simulation output parameters, the first one is a table with two named columns, the first column has a name, 'id' and an integer data type, and the second column has a name 'pot' and a float data type. The second output parameters have a name 'numiter' and integer data type. ODESSI uses numpy [139] arrays for managing tables and arrays. So the 'pots' will be returned to the scientific method that requested the solution as a numpy array that can be used in evaluating an objective function, computing a metric, performing visualization, or any other kind of analysis.

```
sim_dvalues = {'geom':'some_geom_file.txt', 'tol':.0015, 'maxiter':300,
                'brain': .25, 'skull': .018, 'scalp': .44}
```

This line sets the default values of the input parameters. The default values are passed to the simulation when the simulation started, and then only the parameters that are changed from the previous execution are sent to the simulation. Further, any missing parameters in the request for a solution take its default value. Similarly, the upper, lower and precision values are defined for every atomic parameter. Then these specifications are passed to the simulation constructors to create a simulation object.

```
simobj = Simulation(name = sim_name, input= sim_inputs_types,
                     output= sim_output_types)
```

Although these specifications can be specified in the investigation script directly as shown above, in section §7.4 we show that they need to be specified only once and stored in the domain database for reuse. Also, a Traits-UI GUI interface is used to edit these specifications.

7.2.2 Simulation Manager

The simulation manager class is the user interface to the execution manager class. A simulation manager object is created in the main thread by the investigation script as shown in Figure 7.1. A simulation object and other optional features are passed in at the initialization. When a simulation manager object is created, it spawns a thread and starts the execution manager. The simulation manager class currently provides three methods, *stop*, *start* and *get_report*. The get_report method provides some information about the execution, such as the average execution time, the number of reused solutions and the number of actual solutions. More features can be added to this class to monitor the execution of the simulation.

The execution manager manages and controls a single simulation. It acts as a server that provides the simulation response given input parameters. The execution manager is an internal class and is not modified by the user. Typically it gets instantiated and runs in its own thread by the simulation manager and the Planner base class requests solutions from the execution manager.

7.2.3 Execution Manager

Threads request solutions from the execution manager by delivering messages to its mailbox. The Simulation manger extracts a message from the mailbox, handles the request and then delivers the solution to the requested thread mailbox. In handling the request the simulation manager employs a pool of workers for simulation execution and a solution cache database manager thread that maintains a history of solutions for reuse. The interaction mechanism between the simulation manager thread, the database manager thread, the workers and the requesting processes is described below.

1. If the next message in the mailbox is a solution request, the execution manager thread delivers the request to the DB manager mailbox to query if the solution is already exist and it proceed to handle next message in the mailbox.
2. If the message is a successful solution from a worker, then it delivers a copy of the message to the database manager mailbox to update the database with the solution and delivers a copy to the requesting process mailbox.
3. If the message is a query success from the database manager, then it delivers the message to the requesting process.
4. If the message is a query failed message from the database manager, then it delivers the message to the least loaded worker process mailbox.

5. If the message is a quit message from the main thread, then it delivers a quit messages to the workers and database manager and waits until they quit and it quite.

Dynamically spawning new workers or terminating workers for load balancing can be implemented in a straightforward manner. If the number of pending requests is large, then it spawns new workers if there are available resources. If some workers are idle for some time then it terminates some of the workers. Fault tolerance can be implemented easily, for example if a worker is timed out then the request can be resubmitted to another worker.

7.2.4 Workers

The execution manger starts a pool of worker threads, where each worker in the pool corresponds to the execution of a simulation on a resource. For the workers to be able to interact and control a simulation, a slight modification to the simulation interface is required. This can be achieved either through a very simple modification to the main function of the simulation or through a wrapper around the unmodified simulation. The communication protocol between ODESSI workers and the simulation is very simple, where the simulation (or simulation wrapper) responds to four messages. The "param" message allows parameters to be specified for a run. The "go" command initiates a simulation run. The "result" command requests outputs be retrieved from the simulation. The “stop” command asks the simulation to terminate. These commands take the following format:

```
[param]  [input-parameter-name]  [input-parametr-value]
[go]
[result] output-parameter-name
[stop]
```

We chose this approach instead of passing the parameters through the command line and make a system call because in typical scientific domains, the simulation needs a large input files. Therefore, restarting the application every time a solution is requested can be inefficient especially if we conduct investigations that need thousands of simulation runs with the same input data file set and only some parameters needs to be varied as our domain problem.

The worker maintains information about the current state of the simulation parameters. It only sends the parameters that are different from the current state parameters of the simulation. Normally scientific investigation methods vary a subset of the parameters and assume the rest of parameters to take the default values. The workers update the missing parameters with their initial values as specified in the investigation script.

Currently ODESSI support three types of communication with the simulation. When the simulation runs locally the worker communicate with the simulation through pipes. When the simulation runs on different cluster, but on the same file system, the named-pipes are used. In this case the worker creates the fifo files and pass their name to the simulation as argument. It removes these files before it quit. When the simulation runs on remote site, the worker communicates with the simulation through socket. This is implemented by running a simple server on the remote cluster, the server starts the application upon the worker request, and then the server forward all the commands to the application and returns all the responses from the simulation to the worker.

Also, ODESSI supports executing a simulation on a Portable Batch System (PBS). In this case, a multi-threaded server runs on the front node and accepts connections from ODESSI workers. Each connection handled by a thread to service one ODESSI worker. For each connection, the front server handler thread generates a shell script to run an inner server on the inner nodes and submits it to the PBS scheduler. After submitting the shell script the handler thread listens on a port number for connections. The port number is passed to the inner server as an argument. Then the inner server connects to the front server on the provided port number and provides it with its address and port number. The front node thread that handles the connection connects to the inner server and request running the simulation on the node. The inner server runs the simulation on the inner node and interacts with the simulation through pipes. The threads on the front server use select to communicate the parameters and the solution between the inner server and the ODESSI worker.

Remote servers can be started and terminated by ODESSI using utility such as pexpect. Data files can be moved using scp and pexpect as well. Currently we use ssh with public and private keys to start the remote server.

7.2.5 Database Manager

The database manager manages a repository where simulation solutions can be obtained. When the database manager is started it creates a table in the ODESSI database. The name of the table is the simulation name if it doesnt exist in the database. The table fields correspond to the input and output parameters as specified in the investigation script and their names are exactly the names of the parameters. If some parameters are data files, then only the name of the data file is stored. If a table with the same name already exists, then it compares the fields of the table with the input/output parameter names. If they are the same, then it uses that table. Otherwise, it generates a new table. The message interface is very simple for the database manager, composed of query and update messages corresponding to data lookups and additions.

The database manager retrieves messages from its mailbox. If the message is a query message, it updates the missing input parameters in the message with their corresponding initial values and then it looks up the data base. If the query is success then it delivers the solution as success to the mailbox of the execution manager. Otherwise it delivers a query failed message.

7.3 Scientific Data Models

A data model is an important part of any problem solving environment. The main purpose of scientific data models is to manage, share and archive domain data and to provide support for software application for investigation and visualization. Scientific data may come from a variety of sources (e.g., remotely sensed data, experiment, simulation) in a variety of formats which includes multi-dimensional arrays, tables, and scalar and vector fields. Even from a single data source, multiple data sets can be obtained. Each data set typically contains several independent variables (e.g., time, spatial, spectral) and many dependent variables. Therefore, different domains typically have different data management requirement and thus several data models exists to satisfy these needs. Examples of domain specific models includes GML (Geography markup Language)[78], GRIB(Grids in Binary) and many other XML based languages, such as Chemical Markup Language (CML)[37], Molecular Dynamics Markup Language[130], Micro-Array and Gene Expression Markup Language (MAGL-ML)[124], Genome Annotation Markup Language , Numerical Data Markup Language (NDML)[134], Protein Extensible Markup Language (PROXIML) and many more. More recently, there exist several domain independent data models includes, HDF (Hierarchal data Format) [93], CDF[30]/netCDF[137, 160] (Common Data Format), FITS(Flexible Image Transport System)[62] , XSIL (eXtensible Scientific Interchange language)[219]. The common goal of most scientific data models is how to present and manage multidimensional arrays, tables of records, images and their associated metadata.

The traditional approach of handling scientific data is through data file structure. This approach is generally inefficient in storage, access, ease of use, data sharing and interoperability across applications or platforms, in particular, for large and complex data sets. Further, the extensibility of such approach is limited. Any process that accesses the data requires developing software that can manage the arbitrary data. While a reliable investigation environment should support the file approach for rapid prototyping and analysis, the environment should support some type of modern data models for accessing domain data for the purpose of building domain investigation environment. To handle this issue, there should be an abstract layer that interfaces between the data and the investigation tools where arbitrary data objects can be created and managed, and new investigation tools can be added independent of the domain.

Therefore, in the context of scientific investigation environment there is a need for a data model that reflects the structure of the data and how the data can be found, selected and accessed. Building such a reliable scientific investigation environment requires some kind of database model that realizes modern database management system, but is able to handle scientific data sets and applications. It should be easy to use, support large data sets, accommodate multiple data structure and be extensible where new data structures can be added. Such a domain data model should provide a simple way for accessing self-describing data and should handle multiple simulations and application requirement such as investigation, visualization and data analysis. Also, the data model should be independent of the scientific domain. Modern data models put a considerable attention on metadata support for the management of data. Metadata provide the mechanism to formulate queries to select a data object of interest for analysis and investigation. Metadata can be classified into four categories according to their use:

- **System metadata.** The data that describes the structure of the data. It provides information on how to access the data structures such as data types, dimensionality, shape, endianess and size.

- **Attribute metadata.** A user defined metadata that provides information about the data object itself. It is sometimes referred by the data object attributes which is used to find and select the data object.

- **Relationship metadata.** A user defined metadata that provides information about the relationship between a data object and other data objects in the form of references.

- **Descriptive metadata.** A user metadata that provides further description about the data object, such as comments from the user who collected the data. This kind of data is typically in the form free text, for instance, to describe how the data is collected or what can be done with the data.

Relational database management systems provide a modern successful solution for data management in business domain. However, due to the structural nature of scientific data, they do not provide an effective solution. The relational model does not handle multidimensional, hierarchical structures that are common in scientific data sets efficiently. Further, relational databases do not provide sufficient performance for the size, complexity and computational requirement of scientific data. On the other hand, relational database models are successful in metadata management. Therefore, it is sometimes used to manage the metadata while keeping the actual data in other form (e.g., data files).

In summary, we need a data model with the associated software that allows finding and selecting data objects from a domain data sets easy. The data object should be self describing which allow accessing and sharing the data for scientific investigation, data analysis and visualization. The goal is to simplify the data management part of the scientific investigation which allows scientist to focus on the science aspect of the investigation. The HDF5 (Hierarchical Data Format)[**?**] data model and its Python PyTables API[159] provide a powerful data model that allows us to store other domain investigation constructs such as simulation description, resources, and simulation runs provenance. In the following subsection we describe the main characteristics of the HDF5 data model and PyTables packages. The data management component of ODESSI with a scope to provide an easy way of constructing a domain specific scientific investigation environment is described in section §7.4.

7.3.1 Hierarchical Data Format (HDF5)

The Hierarchical Data Format (HDF) is a self-describing file format with a software library that provides an API for storing and retrieving datasets with their associated metadata information. HDF5 was developed for transfer of various types of scientific data among heterogeneous machines and designed to address data management in science and engineering. It provides access to basic atomic and composite data types and simplifies the file structure to include two types of object, Datasets and Groups. Datasets are multi-dimensional arrays of records where the elements are based on atomic and composite data types. The data array can be extended in all possible directions along all dimensions. The size of a data set is not required to be known in advance and data can be written in smaller blocks. Groups are container structures which can hold other groups and datasets. HDF5 files start with a root group, and objects are identified by their pathnames with respect to the root similar to UNIX directory structure. This result in a hierarchical data formats.

Metadata associated with a group or a data set is stored in the form of named attributes (name/value pair) attached to the HDF5 object. Metadata information including endianess, size, shape, architecture and user defined attributes are always stored with the data. More complex structures representing complex data including images and tables can be defined using the HDF5 basic objects: datasets, groups and attributes. Further, HDF5 provide dataspace objects that represent selections over dataset regions. HDF5 index table objects efficiently using B-trees, which make it, works well for time series data (e.g., EEG records, network monitoring). The user of the HDF5 data format does not need to know about the technical low level of the data representation. The user needs to operate only on the higher level structures.

HDF5 has a powerful, simple, efficient and flexible data model that supports files larger than 2 GB with unlimited size of objects stored in them and parallel I/O. The data is represented internally in memory and externally on storage device. The goal of the internal representation is to maximize performance while the goal of the external representation is for data sharing, compact storage, and efficient I/O. The HDF5 library stores compressed and uncompressed data sets. It is designed to take advantage of the power and features of today's HPC computing systems.

For these reasons, we chose HDF5 file format model to store domain data objects and other domain objects (e.g., simulation description as described below) to enable building a domain investigation environment.

7.3.2 PyTables

PyTables [159] is a Python object-oriented package built on top of HDF5 library and numpy package [139] for managing hierarchical and large data sets.

PyTables doesn't provide a complete wrapper of the HDF5 library. However, it provides an efficient, flexible and easy to use tool to manipulate large data table and array objects in a hierarchical data organization. A PyTable table is a collection of records whose records are stored in fixed-length fields. PyTables array objects are analogous to tables with all of their components are homogeneous. They can be extended along single dimension and the array rows can have variable length. Pytables data can be retrieved and post-processed with another HDF5 application and so the HDF5 interoperability and sharing features are preserved.

PyTables support several features includes, 1) variable sized tables and large number of rows, 2)multidimensional and nested tables cells (a table column can be made of other columns), 3) table indexing for efficient search, 4)support for numerical arrays (numpy, numeric, and numarray), these objects are generally used by many data analysis and visualization tools as well as ODESSI, 5) enlargable arrays and access slices of the data sets, 6) Support of a hierarchical data model, 7) Support of user defined metadata, beside supporting system metadata, 8) the ability to read/modify a large portion of generic HDF5 files objects, 9) Support data compression and high performance I/O and 10) Support of files larger than 2GB with architecture-independent format.

7.3.3 ODESSI data coupler

In this section we describe the domain data coupler to ODESSI environment as shown in the design diagram. Domain data objects are defined in terms of a data object class. Currently ODESSI support two types of data objects, multi-dimensional array objects and table object. A variety of complex scientific data structures can be defined in terms of these structure. The multi-dimensional array has a homogeneous data types while columns in the table object can have different data types. Data objects are instances of the DataObject class. Each data object is defined in terms of five attributes:

Kind attribute. The kind attributes are domain controlled vocabulary used to annotate the domain data objects with semantic information. It is used in the search for a particular object. More sophisticated approach through the use of ontology and sematic query in the context of conceptulizing the domain can be used. However, the ontology and sematic queries are not the focus of this dissertation and we leave this area as a future extension to ODESSI environment.

Numerical data. Numerical data (e.g., array or table) is implemented as a numpy array object. The system metadata that provides information about accessing the data structure (e.g., data types, shape, endianess) is stored as part of the numpy array object and they can be qured using the numpy package.

Metadata attribute. The user defined metadata is a dictionary of (name/value) pairs attach to the object. A data object of a specific kind can be located by formulating a query in terms of the objects' metdata.

Parameters list. Each data object can have a list of variables that can be varied to conduct scientific investigations to study their effect on the simulation output. This approach is necessary as generally the set of variable depends on the data object. One example from our head modeling domain, is that the scalp potential dependes on the head tissue electrical properties and so the head tissues properties can be varied to study their effect. The number and kind of head tissues can be different from one geometry to another as segemented from MRI image. For example, for the same subject we could have a geometry that consists of three tissues (skull, brain, scalp) or four tissues tissues (adding csf). Similary, in studying the propagation of light in biological material, the set variables depends on how the tissue under study is segmented into a number of hetrogeneouse segments, each with different optical properties (e.g., refraction and absorption coefficients for each seqgment).

Reference. The data object location in the domain data hierarchy, this reference is used to get the object from the HDF5 file. The refernce is similar to the UNIX directory path name.

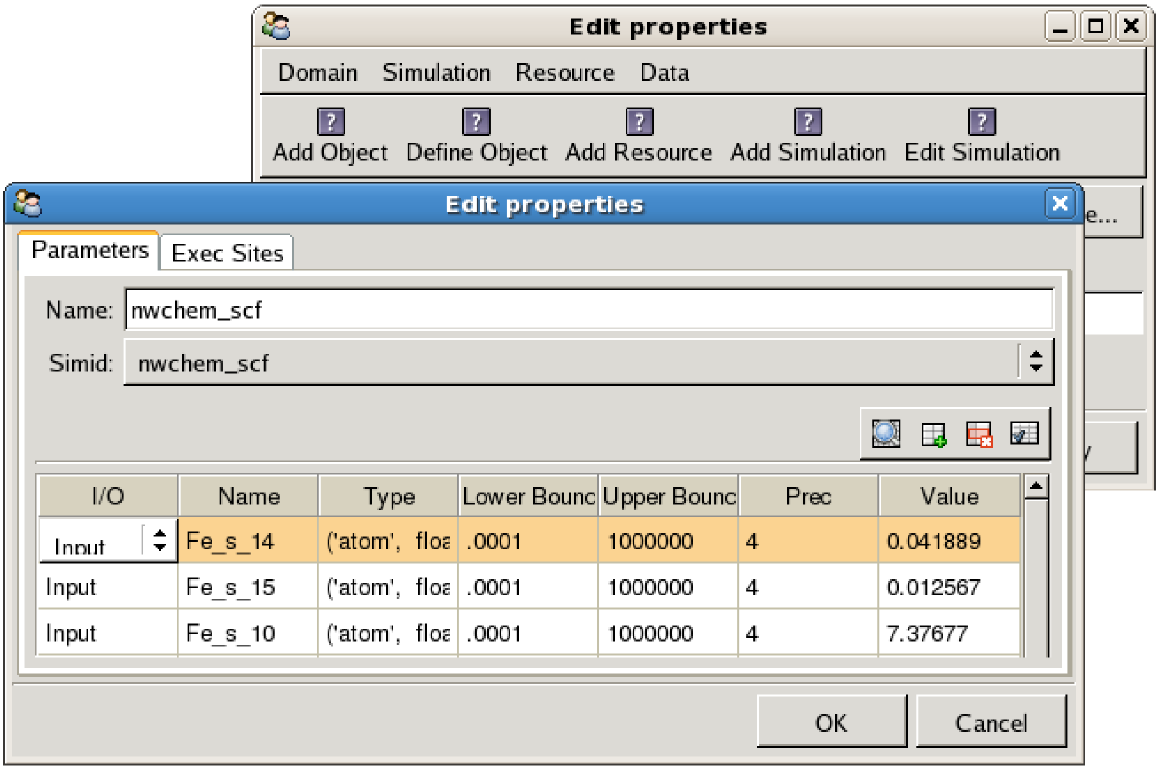
Since ODESSI interacts with a simulation through files or pipes in a loosly manner, the data object (array or table) must be written to a file, and then the file copied to the execution directory, or passed through a pipe to the simulation. Since (in general) there is no standard way of the format of the scientific data file object expected by simulations, the data object must be translated into a form that is expected by the simulation. ODESSI allows the user to provide a translation function that translates the data object to the form accepted by the simulation as explained in the simulation coupler section. Alternatively the user can provide a wrapper function to the file prduced by the generic methods provided by data object class. The data object class provide two generic methods to read/write data objects from/to a data file. When the write method writes a data object, the file consists of a header section and a data section, the header section contains the metadata and parameters and the data section contains the actual data.

Domain data objects are stored in an HDF5 file. Each data object is stored as a leaf in the HDF5 hierarchy. The Metadata, kind and the parameters list are stored as attributes to the data objects.

7.4 Building a Domain PSE

Several scientific disciplines are now focusing on developing domain environment for scientific invetigation. The focus of these environment is on how to allocate and coordinate the use of computational resources (both system and software) to create runs, manage scientific domain investigations, manage the domain data, and experiments results in an easy to use environment. Environments such as ViroLab[210], NEES[135] and BIRN [24] address some of these issues by offering higher-level services (e.g., analysis, data management, simulation) while hiding backend complexity. The ability to abstract and reapply scientific methods for new scientific investigations or new scientific domains in these environments is not supported. The scope of ODESSI framework is to enable building such domain investigation environment by abstracting the common components of such environment in a reusable framework that can be applied on other domains. This includes simulation execution, domain data management, applying common scientific methods and procedures, and investigation provenance.

In ODESSI the scientific domain is presented as a HDF5 file, where the simulation description, data objects descriptions, and simulation runs are stored and managed. The ODESSI framework provides the simulation execution engine, the common scientific methods to be applied in the investigation, and the domain data managements. In addition, ODESSI can be extended with other domain-specific methods and procedures. The HDF5 file format guarantees interoperability with other applications, platforms and languages. ODESSI specific objects (objects that are meaningful only to ODESSI) in the HDF5 can be ignored by other application. In the following we describe the domain management component of ODESSI which includes simulation description, data objects and other domain related objects user



Name: nwchem_scf

Simid: nwchem_scf

I/O	Name	Type	Lower Bounc	Upper Bounc	Prec	Value
Input	Fe_s_14	('atom', floa	.0001	1000000	4	0.041889
Input	Fe_s_15	('atom', floa	.0001	1000000	4	0.012567
Input	Fe_s_10	('atom', floa	.0001	1000000	4	7.37677

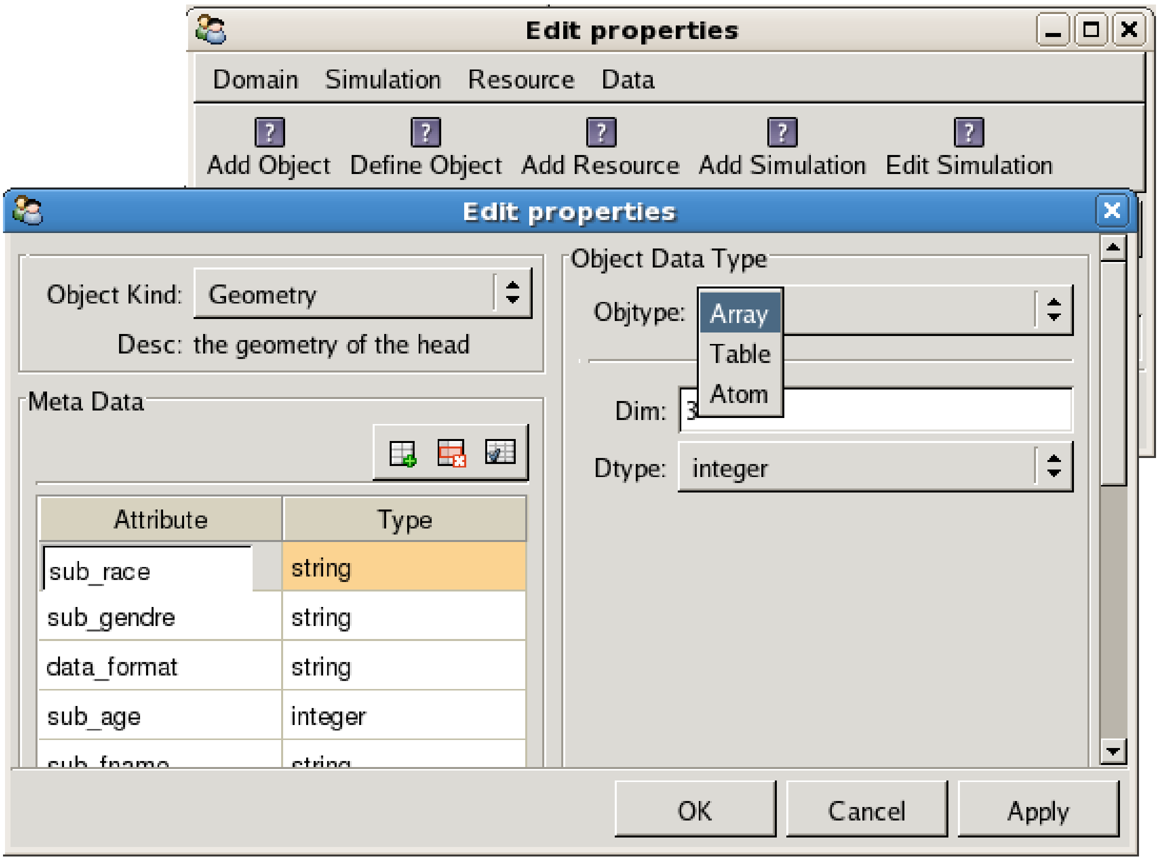
FIGURE 7.5: The simulation description editor: The name of the simulation, its IO parameters and the execution sites where the simulation is available for execution are specified.

interface. All domain objects are specified and edited using Traits-UI editors, which does not require the user to have any experience with HDF5 API or PyTables ABI. The domain HDF5 file is organized in four groups under the root group and managed by ODESSI domainDB class as described below.

Simulations group. This group is a container that contains the descriptions of the domain simulation objects as described in section §7.2.1. Each simulation object is uniquely identified by a simulation id which is used to retrieve the simulation from the domain database for execution in the investigation script. Figure 7.5 shows the Traits-UI editor provided by the domainDB manager. The editor is used to specify the simulation executable, its input/output parameters and their data types, lower/upper bounds, and default values. In the second tab of the editor the user can specify the executable binary path, working directory, and the environment variables on each resource. The simulation menu provides other functionality for editing, deleting and adding simulation objects. Alternatively, the simulation object can be defined and manipulated using the domainDB methods into the investigation script as well.

Data group. Under this group, the domain data hierarchy is stored. The ODESSI domainDB manager provides a method to define a data object schema. The data object schema (implemented as a Python class) defines the object kind that labels the object with semantic information (e.g., geometry, potential, temperature). Also, the data object schema defines the metadata associated to all objects of the same kind. A query about a data object is formulated in terms of these metadata. Each data object schema is uniquely identified by the kind attributes (no two schemas can have the same kind). Figure 7.6 shows the Traits-UI editor used to define and edit data objects schemas (instance of the data object schema class).

Once the schema of a data object is defined, data objects with the kind defined in the schema can be added by instantiating the data object or using the data object Traits-UI editor as shown in Figure 7.7. In addition to the schema



Object Kind: Geometry

Desc: the geometry of the head

Objtype: Array

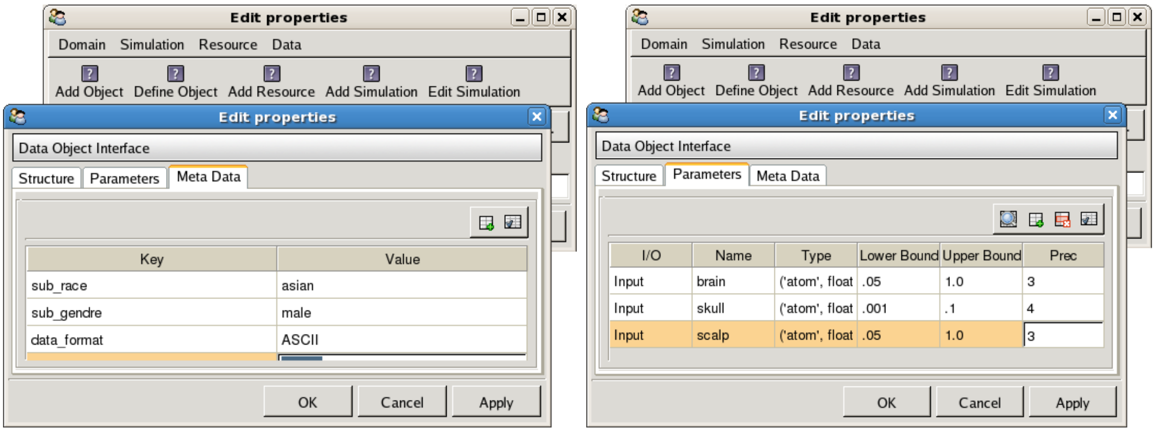
Dim: 3

Dtype: integer

Attribute	Type
sub_race	string
sub_gendre	string
data_format	string
sub_age	integer

FIGURE 7.6: The data object schema description editor, used to define the data object structure and metadata.

metadata, the user can add other metadata associated to a particular object, but these metadata will not be used in the query for an object. They provide extra information about the object. Figure 7.7 (left) shows the metadata editor. In addition, to the object kind and metadata, each data object can have a list of variables. These variables extend the simulation input parameters and can be used to investigate the effect on the simulation output. Figure 7.7 shows the data object parameters list editor.



Key	Value
sub_race	asian
sub_gendre	male
data_format	ASCII

I/O	Name	Type	Lower Bound	Upper Bound	Prec
Input	brain	('atom', float	.05	1.0	3
Input	skull	('atom', float	.001	.1	4
Input	scalp	('atom', float	.05	1.0	3

FIGURE 7.7: The data object editor, used to create and edit a data object for a given kind. The metadata editor (left), and the parameters editor (right).

System resources group. This group contains information about system resources including the machine address, login name, and passwords. Further information about the system resources necessary for executing the simulation can be added in this group.

Invetigations group. Under this group the domain runs and investigations are stored. Each run is identified by a run id. The description of the scientific methods used in the investigation and the simulation id can be stored as part of provenance management.

7.5 Summary and Conclusions

The main aspects of scientific investigation in computational science can be classified into three categories 1) the application of scientific methods includes the specifications and execution, 2) the execution of the simulation and 3) the domain data management. ODESSI achieves the design goals and requirements by factoring out these aspects and then integrates them in a loose manner. This design makes ODESSI environment extensible and open. It is extensible, in the sense that new methods can be added independent of domain data, simulations, or the execution of simulation. Also, the simulation can be specified independent of the scientific methods. It is open, in the sense that scientific methods can be customized to meet the domain problem requirement.

The ODESSI design provides explicit separation between methods' specification and execution. A scientific method is captured in a module of three classes: specification class, interface class, and method logic class. The interface class is concerned with executing and monitoring the scientific method in its own thread. It provides methods to interact with the running scientific method (e.g., to monitor the execution and visualize intermediate results). The specification class allows the user to customize the method. The logic class does the actual computations. All these classes have base classes that provide some basic common functionality. The specification base class allows storing the method specifications with the results as part of provenance management. The interface base class provides methods to launch the method for execution in separate threads and to stop and start executing the method. The logic base class abstracts the interaction of the method from the simulation execution. This separation has other advantages.

For example, it allows building GUI interface to edit the method specification without mixing between the view and the model.

The separation of the execution of the simulation from the scientific method has several benefits. First, adding a new scientific method will not involve any concern of how to execute the simulation on distributed resources which makes the environment extensible. Second, no modification is required on the existed scientific methods when the underlying system architecture changes. Third the scientific method can automatically take advantage of any increase in the available computational power without any modification. The separation between domain data and scientific methods allow developing methods independent of the domain data structure which enable reuse of the methods across several domains and developing general purpose scientific methods.

This chapter provides the main design decisions and technology choices to implement the conceptual design of ODESSI framework that satisfies the main requirement. Scientific investigations can be conducted by executing a scientific investigation scripts that defines the domain data files, the simulations to use, and apply one or more scientific methods. Domain data objects can be obtained from the domain data base by requesting data objects from the domain manager. Scientific methods can be instantiated, customized and executed in their own threads. Multiple scientific methods and multiple instances of the same scientific method can be executed concurrently. The simulation solutions from previous executions can be reused. The next chapter will show how the ODESSI framework can be applied for real scientific investigations.

CHAPTER VIII

ODESSI EVALUATION

In this chapter we used ODESSI to conduct several scientific investigations in two different domains, the human neuroscience domain and the computational chemistry domain. For the computational chemistry domain we used ODESSI to extract the model parameters and to conduct several parametric studies to understand the precision of the model output. In the human neuroscience domain we used ODESSI in tuning the model convergence parameters, extracting the model parameters, studying the effect of the geometry resolution, studying the model output sensitivity to several input parameters, and managing domain data.

The investigations in this chapter demonstrate how easy it is to apply ODESSI to arbitrary simulation and conduct various kinds of investigations leveraging HPC. Also, they show how ODESSI integrates the three aspects of scientific investigations: the execution of the model, the management of domain data, and the application of scientific methods. The provided investigation scripts realize the integration of these aspects in conducting scientific investigations. This design factors out the computational parts of the scientific investigation which allows scientist to focus on the science aspect of computational science.

8.1 Computational Chemistry Domain

One fundamental problem in computational chemistry is the calculation of the eletronic structure for a given molecule. The electronic structure is often represented by molecular orbitals, which are determined during the computation of the electronic energy. This is done using a non-linear iterative approach called the *self-consistent field (SCF)* method. Once computed, several properties of the molecule can be calculated such as electron density and electrostatic potential. The molecular orbitals are typically expanded as a linear combination of a *basis set*. A basis set is a set of mathematical functions used to approximate the atomic orbitals of a molecule. In quantum chemistry the calculations are performed within a finite set of basis functions centered at each atom within the molecule.

The atomic orbitals that correspond to a set of functions which decay exponentially with distance from the nuclei are called *Slater Type Orbitals (STO)*. They take the following general mathematical form,

$$STO = N \exp(-\alpha r),$$

where N is a normalization constant, α is the orbital exponent, and r is the radius in Angstroms. The computations using Slater orbitals are expensive. To simplify the computation, generally each STO is approximated as linear combinations of *Gaussian Type Orbitals (GTO).* There is no major difference in these two methods when computing the molecular orbitals of small molecules. But major discrepancies occur for larger molecules of 30 and more atoms. GTOs are computationally efficient, and typically lead to a significant computational saving but less accurate. GTOs take the following mathematical form,

$$GTO = 2\xi/\pi^{(.75)} \exp(-\xi r^2).$$

Today, there exist hundreds of Gaussians-type basis sets. The smallest set of these functions required to represent all electrons on each atom (such that for each atom in the molecule a single basis function is used for every orbital) is called the *minimal basis sets.* The accuracy of the GTO approximation depends on the number of Gaussian functions used in the approximation. The more Gaussian functions used, the better GTOs approximate the STO. It is desired to determine the optimal exponent coefficients ξ of the Gaussian functions that minimizes the SCF energy of the molecule.

We used ODESSI to determine the optimal exponents, ξ, of four Gaussian functions that minimizes the SCF energy of the iron atom (Fe). It is straight

Listing VIII.1: NWChem input file(left) and its parameterization (right)

```
Title ``Calculation of Fe''
Start Fe_basis
echo
charge 0
geometry autosym noautoz units angstrom
 Fe       0.00000       0.00000      0.00000
end

ecce_print ecce.out
basis ``ao basis'' spherical print
  Fe library ``Wachters+f''
  Fe s
      1.00000    1.0
  Fe p
      1.00000    1.0
  Fe d
      1.00000    1.0
  Fe g
      1.00000    1.0
END
scf
  vectors input scf.movecs3
  nopen 6
  uhf
  maxiter 99
end
```

```
def write_nwchem_if(params, nwchem_in):
    nwinput = """
  Title ``Calculation of Fe''
  Start Fe_basis
  echo
  charge 0
  geometry autosym noautoz units angstrom
  Fe       0.00000       0.00000      0.00000
  end

  ecce_print ecce.out
  basis ``ao basis'' spherical print
  Fe library ``Wachters+f''
  Fe s
      %(Fe_s_1)s  1.0
  Fe p
      %(Fe_p_1)s  1.0
  Fe d
      %(Fe_d_1)s  1.0
  Fe g
      %(Fe_g_1)s  1.0
  END
  scf
      vectors input %(movecs)%s
      nopen 6
      uhf
      maxiter %(maxiter)s
  end
""" % (params)
    F = open(nwchem_in, w)
    F.write(nwinput)
```

forward to find the exponents of other functions by repeating the same calculations, once ODESSI has been set up for the problem with iron.

We used the NWChem package [142] to augment the “Wachters+f” basis set with additional Gaussian functions (s, p, d, g). This is done by computing and minimizing the SCF energy with respect to the exponents of the Gaussian functions. This is a good example of how ODESSI can connect with a simulation without the need to modify its source code.

NWChem is a computational chemistry package made up of various functional modules, including SCF energy modules. NWChem package takes an

Listing VIII.2: Wrapper function to run NWChem program

```
def nwparse_out(outfile):
    for line in open(outfile, 'r').readlines():
        if 'Total SCF energy = ' in line:
           return line.split('=')[1].strip()
    return 'nan'

def run_simul(infile, outfile):
    p = os.system("nwchem %s 1>%s" % (infile, outfile))

def nwchem_sim():
    params=dict(Fe_s_1=1.0, Fe_p_1=1.0, Fe_d_1=1.,
                Fe_g_1=1.0, maxiter='99', movecs='scf.movecs')

    nwchem_inf = 'nwchem.in'
    nwchem_outf ='nwchem.out'

    while True:
        line = raw_input().strip().split()
        command = line[0]

        if command == 'stop': break
        elif command == 'param':
            param_name, param_value = line[1], line[2]
            if param_name in params.keys():
                params[param_name] = param_value
        elif command == 'go':
            write_file(params, nwchem_inf)
            run_simul(nwchem_inf, nwchem_outf)
        elif command == 'result':
            outparam = line[1]
            outvalue = nwparse_out(nwchem_outf)
            print 'result', outparam,  outvalue
            sys.stdout.flush()

if __name__== '__main__':
    nwchem_sim()
```

input file that contains directives on how to run a module. An example of a NWChem input file used in the computation of the SCF energy of the Fe atom is shown in Listing VIII.1(left). Typically, a chemist writes the input file that specifies the science under investigation. Then the same input file is used to conduct several investigations by varying parameters to study their influence on the output of the model under the study.

To allow ODESSI to vary these parameters in applying scientific methods, the input file is parameterized by writing a formatted string to a file. Listing VIII.1(right) shows the parameterized input file of Listing VIII.1(left). In this example, we parameterized the exponents of four Gaussian functions (Fe s1, Fe p1, Fe d1, and Fe g1), the coefficients' file name (scf.movecs), and the maximum number of iterations (maxiter). Similarly, the output of NWChem program is a text report that contains a large amount of information. Typically, one needs to parse the output file for the desired information. Since we only interested in the SCF energy, we only need to parse the output file for the SCF energy and return the result.

Listing VIII.2 shows the Python wrapper program used to accept and respond to commands from ODESSI workers. It quits when it receives the command "stop". When the received command is "param", it updates the parameters dictionary with the received parameter values. When the command is "go", it writes the nwchem input file and executes the NWChem program with the input file as argument. When the command is "result" it parses the output file for the SCF energy and returns the SCF energy to the worker. The wrapper on the clusters gets executed by the ODESSI server running on the clusters. The server communicates the parameters and the results between ODESSI workers and the wrapper.

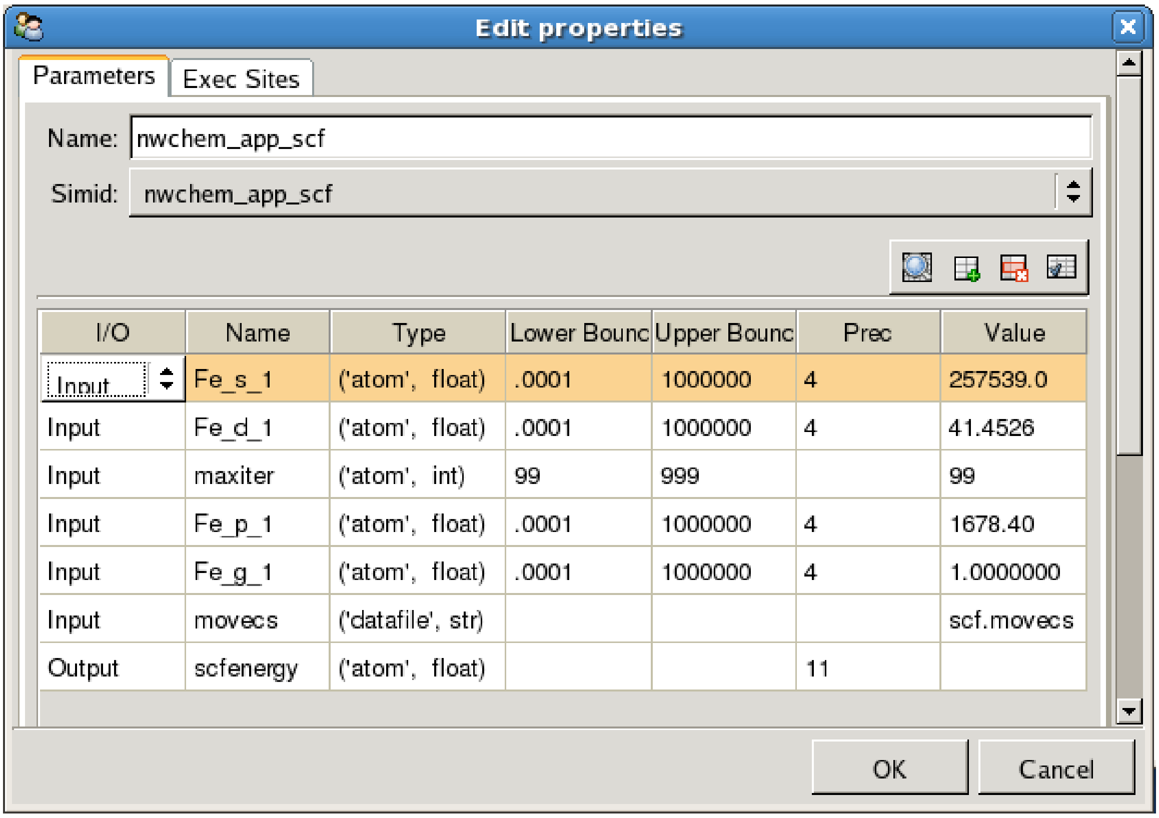
The next step is to specify the wrapper program as the simulation under investigation to ODESSI. It can be specified directly in the investigation script or by using the domainDB manager which saves the specification for later use. By

Listing VIII.3: Invitigation script to run chemistry optimization

```
db = DomainDBInter(database = 'nwchem.h5')
sim = db.getSimulation('nwchem_app_scf')

clusters = ['nic1', 'nic2', 'nic3']
sm = SimmMan(workers=18, sim=sim, useDB = False, clusters=clusters, db=db)

vars =  ['Fe_s_1',       'Fe_p_1',       'Fe_d_1',       'Fe_g_1' ]
lb   =  {'Fe_s_1':0.001, 'Fe_p_1':0.001, 'Fe_d_1':0.001, 'Fe_g_1':0.001}
ub   =  {'Fe_s_1':800000,'Fe_p_1':800000,'Fe_d_1':800000,'Fe_g_1':800000}
prec =  {'Fe_s_1': 3,    'Fe_p_1': 3,    'Fe_d_1': 3,    'Fe_g_1': 3}
init =  {'Fe_s_1':3854.0,'Fe_p_1': 3.9,  'Fe_d_1': 875.0,'Fe_g_1': 987.0}

simout = ['scfenergy']
obj    = SCFEnergy()

op1 = Optimization(method="SA", simman=sm, vnames = vars, vlb = lb,
                   vinit = init, vub = ub, vprec = prec,
                   sim_outputs = simout, objfunc = obj)
op1.go()

op2 = Optimization(simman=sm, sim_outputs=simout, objfunc=obj, custom=True)
op2.go()

print op1.results()
print op2.results()
```

```
class SCFEnergy:
     def __init__(self):
          pass
     def __call__(self,simout):
          objvalue = simout['scfenergy']
          if (str(objvalue)) == 'nan':
              return 1.0
          return objvalue
```

using the GUI interface as described in §7.4 and shown in Figure 8.2, we only need to fill out a form. Under the parameters tab we specified the six input parameters (Fe_s_1, F_p_1, F_d_1, F_g_1, movecs, and maxiter) and the output parameter (scfenergy). For each input parameter, we specified its data type, precision, default value, and its upper/lower bounds. In this example, the data types of the “movecs” and maxiter” parameters are “datafile” and “int”. The data types of the other parameters are “float” (Python float is equivalent to C double). When the data type is “datafile”, ODESSI copies the file to the remote execution site working directory



Name: nwchem_app_scf

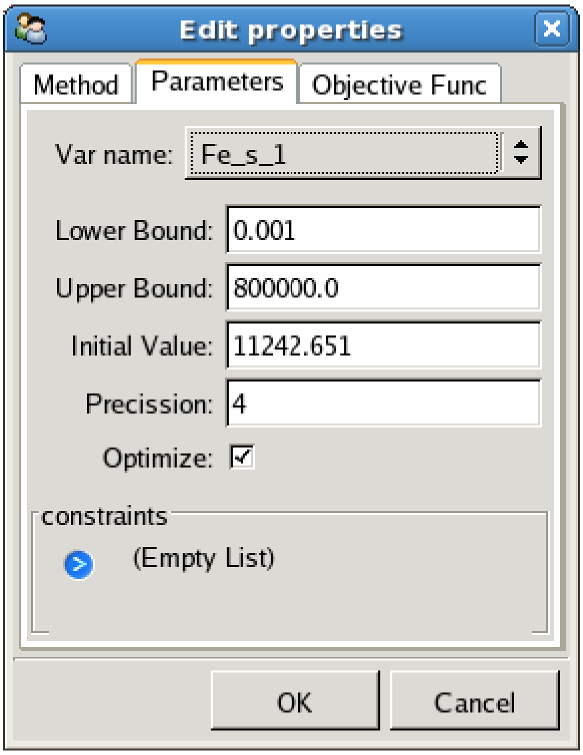
Simid: nwchem_app_scf

I/O	Name	Type	Lower Bounc	Upper Bounc	Prec	Value
Input	Fe_s_1	('atom', float)	.0001	1000000	4	257539.0
Input	Fe_d_1	('atom', float)	.0001	1000000	4	41.4526
Input	maxiter	('atom', int)	99	999		99
Input	Fe_p_1	('atom', float)	.0001	1000000	4	1678.40
Input	Fe_g_1	('atom', float)	.0001	1000000	4	1.0000000
Input	movecs	('datafile', str)				scf.movecs
Output	scfenergy	('atom', float)			11	

FIGURE 8.1: NWChem simulation specification.

and passes its name as a parameter. For the output parameter, we specified its data type and precision. Under the resources tab we specified information about executing the simulation on each resource which includes, the resource name, the working directory name, and the simulation binary path on each resource.

Once the simulation is specified, we are ready to conduct various kinds of investigations leveraging ODESSI's ability to run multiple simulation simultaneously. It should be noted that NWChem is enabled for parallel execution and ODESSI could run the simulations in parallel mode. Listing VIII.3 shows the



Var name: Fe_s_1
Lower Bound: 0.001
Upper Bound: 800000.0
Initial Value: 11242.651
Precission: 4
Optimize: ☑
constraints
(Empty List)

FIGURE 8.2: NWChem optimization specifications.

investigation scripts that we used in conducting optimization study to find the optimal exponents of the four Gaussian functions. In the first line we get the simulation object that we specified before. In the second line, we created a simulation manager object that is used by methods to get simulation solution. In this example, the simulation manager is specified to employ 18 workers that can start and control 18 instances of the simulation on three clusters *nic1*, *nic2*, and *nic3*. Each one is a p655 8-way machine runs the AIX operating system. Workers start simulation instances on clusters in a round robin fashion. So in this example, each cluster runs 6 instance of the simulation. Even though we only have the sequential version of NWChem installed on these clusters, using ODESSI we were

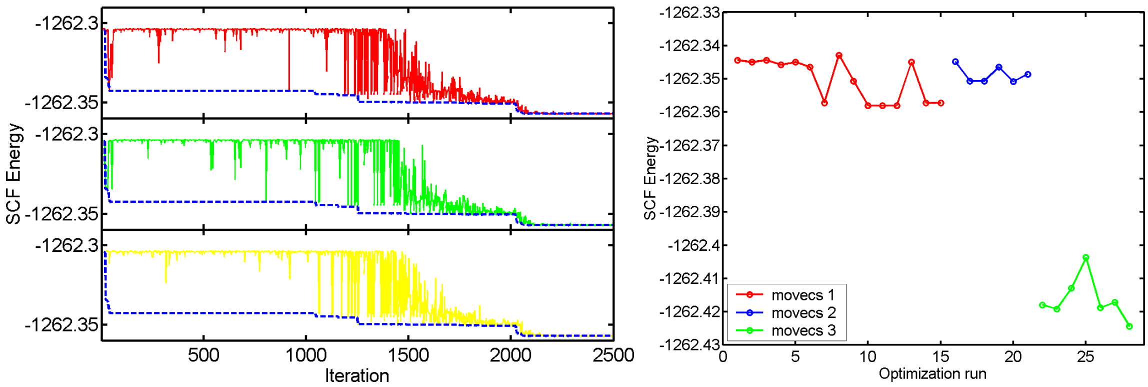


FIGURE 8.3: The dynamics of 3 processes out of 12 processes of the parallel simulated annealing method(left). The optimal objective function values for several optimization runs using different configurations and input coeffecient files (left).

TABLE VIII.1: SCF Energy optimal exponents from several runs

Run	Fe s 1	Fe d 1	Fe p 1	Fe g 1	SCF Energy	coeff.
10	423.70	195.96	6.53	2.0904	-1262.357978968383	1
14	423.65	195.82	6.54	635713.1	-1262.357181644812	1
17	423.64	196.28	11266.22	149955.1	-1262.350603950666	2
20	386.09	3737.05	5.92	3123.3569	-1262.350817304612	2
21	424.33	15736.35	26.25	783696.03	-1262.348615883747	2
28	424.82	195.02	7.041	37607.46	-1262.424367113445	3

able to utilize all the available power in conducting investigations. Actually, if the amount of available power is increased, we can just add more instances of the same method or apply other methods.

In the second section of the investigation script, we specified and started two instances of the optimization module. The two instances run concurrently and share the same simulation manager. In this example, each instance runs the parallel simulated annealing method with a degree of parallelism of 12. This gives a total degree of parallelism of 24 which makes the 18 workers fully utilized even if some of the processes in the parallel method are waiting for synchronization (below we investigated this issue further). The user defined objective function for this example is shown in Listing VIII.3. Since we are only minimizing the simulation output, nothing needs to be passed in the initialization. The objective function returns the simulations' output when it is valid. If the simulation output is invalid (a nan number) then it returns 1.0 which is a too large number since all energy values are negative.

We ran the investigation script several times experimenting with different configuration of the simulated annealing algorithm, different initial values, and different coefficient files. Figure 8.3(left) shows the dynamics of three out of 12 processes of the simulated annealing algorithm. Figure 8.3(right) shows the optimal values of the SCF energy obtained in several runs, each run with different configuration parameters. The optimal exponents corresponding to some of these runs are shown in Table VIII.1.

The study in this section demonstrates how easy it is to set up ODESSI to work with an arbitrary model and once done how easy it is to conduct scientific investigations leveraging distributed computing without any requirement on the user side.

8.2 Human Neuroscience Domain

The ultimate goal of our research in head modeling is to estimate the locations of the active brain regions given measured electroencephalogram (EEG) recordings. Called the *source localization* problem, its accurate solution will provide an opportunity to analyze cortex dynamics at high temporal and spatial resolution. To review, the source localization problem has two parts:

1. *Forward problem*: given electrical sources (e.g., cortex dipoles), tissue geometries and conductivities, determine head volume and scalp electrical potentials.
2. *Inverse problem*: given an accurate forward solution, find optimal sources to match measured scalp potentials.

An accurate forward solver requires knowledge of the head tissues geometry (obtained from MR or CT images) and their conductivities. To determine the conductivities of the head tissues, we must solve the conductivity inverse problem. Here, a small current is injected in a subject's head and the response is measured on the scalp using bounded Electrical Impedance Tomography (bEIT) technology. A search for optimal conductivities parameters can then be performed using the forward simulation compared to the measured potentials. Once the conductivities are found for an individual, a distributed dipole linear inverse solver can be built for EEG localization V.

There are several challenges in this research. From the start, the source localization problem is ill-posed, since EEG measurements are made on (up to) 256 sensors and there may be thousands of cortex dipoles active. In addition, there are multiple sources of measurement error and modeling uncertainties that ultimately contribute to the accuracy of the solution as well as the performance. Measurement errors include the quality of MR/CT images, electrode and dipole registration, injected current level, and the EEG electrode data. These errors lead to modeling inaccuracies which propagate uncertainty in the solution results and also can affect computational efficiency. These include discretizing the PDEs, adjusting the computational grid resolution, and accurately segmenting the head tissues. Further, selection of parameters and modeling algorithms in the forward and inverse solvers also influence the final result.

How can we understand the quality of our source localization solutions and their use in dynamic brain analysis when dealing with multiple sources of measurement error and modeling uncertainties in constructing electromagnetic head models? Our desired scientific investigations involve computational processing and simulation to generate candidate models, as well as verification and validation to determine the effects of uncertainty and the robustness of solutions.

8.2.1 Simulation and Domain Data Objects Specifications

Although ODESSI allows the user to specify the simulation, data objects and scientific methods directly in the investigation scripts, it is more efficient and productive to save these object in a domain database to be reused and shared with other user (e.g., the lab who supplies the data). The domain database management component of ODESSI, described in detail in §7.3, is used for this purpose.

First we created a head modeling domain database managed by the ODESSI domainDB manager and then we defined three data object schemas. The first schema defines the geometry data object, the second schema defines the sensors data objects and the third one defines the measured data objects. Different labs provide these objects. The image processing lab, for instance, provide the geometry after processing the MRI/CT image. The EEG lab provides the sensor positions and the current injection data. One goal of the domain database component of ODESSI is to allow different labs or researcher to update the domain database with out human interaction. Once these schemas are defined, users can add data objects by filling out a form. In the geometry schema we defined the kind attribute to be "Geometry". For the sensors schema the kind attributes is "Sensors", and "CInjection" is the kind attribute for the data sets obtained from current injection data. The schema metadata attributes are used to query objects from the domain database. Data objects can extend the schema metadata attributes. However, the extended attributes are object-specific metadata and can not be used in the query for an object. Then we specified the simulation using the simulation user interface. Once the simulation object is specified and the domain data objects are added to the domain database, we are ready to conduct several investigations using different objects from the domain database.

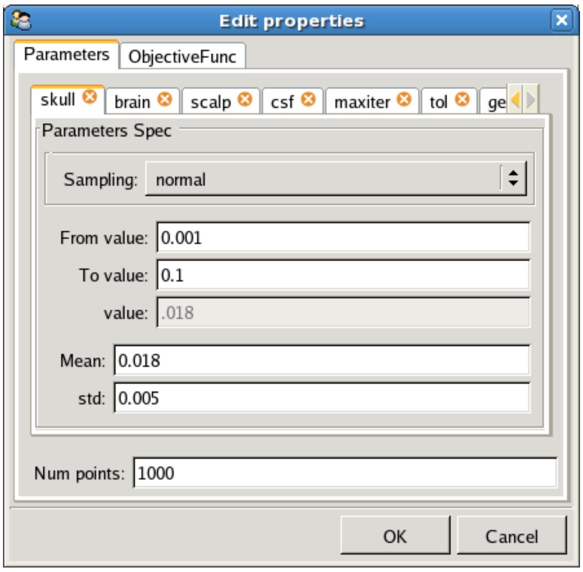
Listing VIII.4: senstivity analysis script

```
#specify the domain db
db = DomainDBInter(database = 'head_modeling.h5')

#get a domain model to be used in the analysis
sim = db.getSimulation('ADI')

#define a simulation manager object, that uses 16 workers
sm = SimMan(workers=16, clusters=['mist'], sim=sim, db = db)

#get the geometry from the domain db
geom = db.getObject('Geometry', query=''fname=='SomeName' and gresolution==1'')
sim.setParam('geom', geom)

#get a sensors net that is registered to the geometry
elecs = db.getObject('Sensors', query=''geometry == '%s' '' % geom.name)
sim.setParam('sensors', elecs)

#define sampling for each parameter
s1 = SParam(name='skull', fromvalue=.001, tovalue=.1,   dist=('normal',.018,.005))
s2 = SParam(name='brain', fromvalue=.05,  tovalue=1,    dist=('normal', .25, .06))
s3 = SParam(name='scalp', fromvalue=.05,  tovalue=1,    dist=('normal', .44, .13))
s4 = SParam(name='csf',   fromvalue=1.4,  tovalue=2.2, dist=('normal', 1.78,.13))
s5 = SParam(name = 'maxiter', value=800)
s6 = SParam(name = 'tol', value=0.00015)

sens = Senstivity(params=[s1, s2, s3, s4, s5, s6], simman=sm,
                  num_samples=1000, custom=True)
sens.go()
sens.results()
```

8.2.2 Sensitivity Analysis

In head modeling it is important to understand the sensitivity of the scalp potential to a variety of model uncertainties. One important analysis is to study the sensitivity of the scalp potential due to the conductivities of the head tissues. Quantifying this sensitivity can help in prioritizing the research and reducing the number of model parameters in other studies (e.g., optimization).

In this section we applied ODESSI for regression analysis to study how the uncertainty in each electrode potential can be apportioned to uncertainties in the inputs. In this analysis, we only considered the head tissue conductivities. A



Sampling: normal

From value: 0.001

To value: 0.1

value: .018

Mean: 0.018

std: 0.005

Num points: 1000

FIGURE 8.4: Head modeling sensitivity analysis specifications

multivariate sample of 1000 points of the head tissue conductivities is generated. The conductivity of each head tissue is sampled from the normal distribution with mean equal to the average accepted value from the literature and the standard deviation is chosen such that the distance between the mean conductivity and the lower and upper bounds is about three standard deviations. Listing VIII.4 shows the investigation script we used to conduct this analysis. The parameters can be specified either through the script or using the UI interface as shown in Figure 8.4. With ODESSI it is easy to repeat this analysis for different configurations. Figure 8.5 shows the results of the analysis using three different current injection pairs and two different geometry resolution ($1mm$ and $2mm$).

Figure 8.5 shows distributions of the electrodes sensitivity due to changes in tissue conductivities using different configurations. Positive sensitivity coefficients

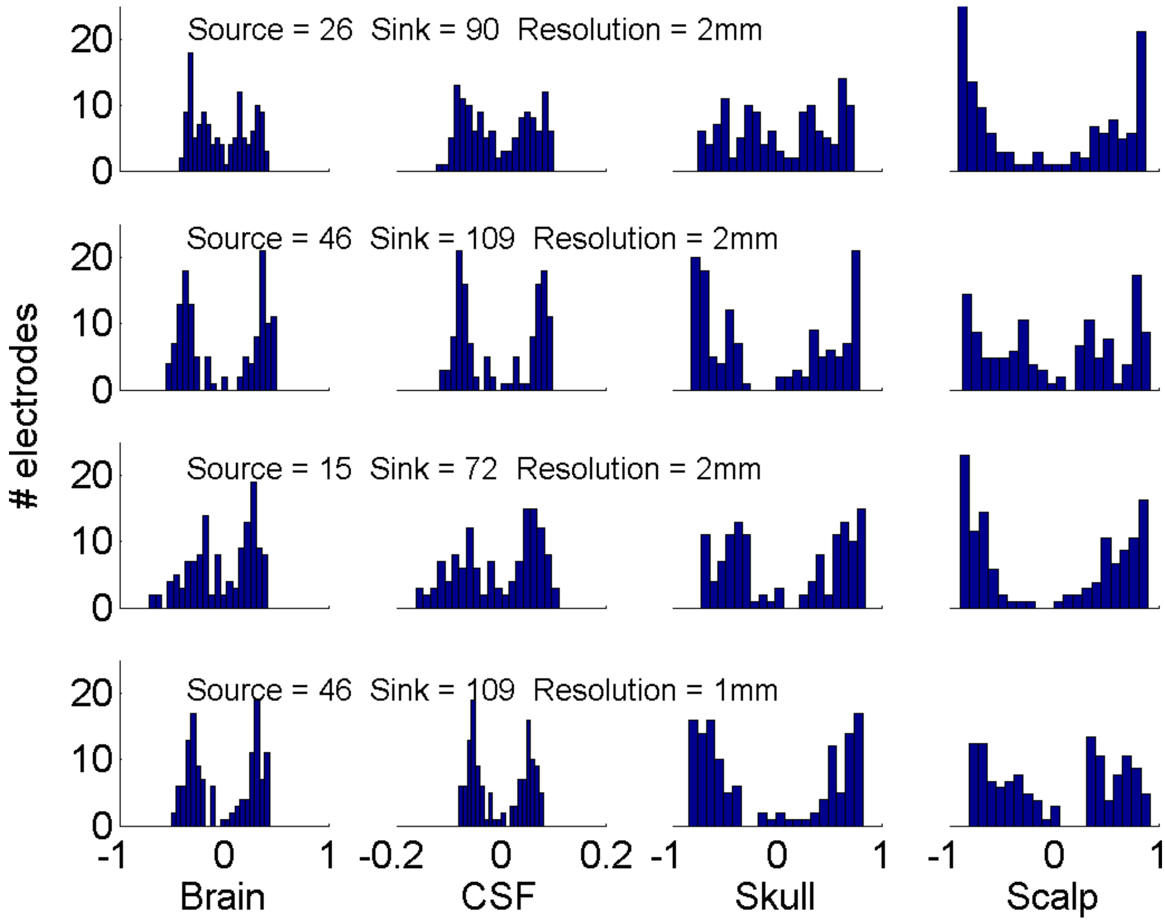


FIGURE 8.5: Distribution of the electrode's sensitivity due to changes in the Brain, CSF, Skull and Scalp tissue conductivities using several configurations.

correspond to electrodes near the current source while negative coefficients correspond to electrodes near the sink. From the distributions we see that the potentials at all electrodes are insensitive to variation of the CSF tissue. This can be reasoned to the fact that the CSF tissue size is small and the variation in its conductivity is small. The second important observation is that the potentials are sensitive to changes of the brain conductivity. This observation contradicts the belief that most of the current is shunted in the scalp. Therefore, we believe it is possible to explore the brain with EIT technology. We explain this sensitivity due to the fact that the brain is a big tissue and the holes in the skull –which our forward solver captures– allow the current to go through the skull (contrary to spherical models). The third observation is that the potentials are highly sensitive to changes in the skull and scalp conductivities as expected, since the current sources are on the scalp. These observations are confirmed using low/high geometry resolution and different current injection pairs. Identifying and ranking the sensitivity of the electrodes due to model input variables are very important in our research. For instance, the conductivity of the CSF tissue can be considered homogeneous and be fixed at the literature accepted value. Also, the contributions of the electrodes potentials in computing the objective function can be weighted based on their sensitivity in the conductivity inverse problem.

ODESSI made conducting this analysis simple, which allowed us to experiment with different geometry resolution and different current injection pairs to build confidence of our results and make solid conclusions. We parameterized the sensitivity testing template in ODESSI and interfaced the simulation code. Once configured, the investigation required thousands of simulations to generate the results. These simulations were fully automated by ODESSI.

8.2.3 Conductivity Modeling – Optimization

In our early programming of the conductivity inverse problem, we used a simplex search method with an isotropic forward model. We soon realized that the simplex algorithm was not robust enough for our problem, and the isotropic solver lacked required precision. In both cases, the code had to be modified to incorporate improved methods. With ODESSI, we were able to set up the problem, conduct and adjust the optimization parameters by only interfacing with the optimization method module. In addition we were able to conduct other investigation.

Listing VIII.5 shows the investigation script we used to conduct optimization studies to extract the conductivities of the human head. In the scripts, we first specified the domain database where domain data and simulation specifications are stored and can be retrieved. Once we get all domain data objects from the domain database, we created a simulation manager object which is running in its own thread. The simulation manager object can use any available distributed resources to run the simulation and return solutions to methods who requested the solutions. Since each data object has a set of associated parameters that can extend the simulation input parameters, the simulation list of parameters will be extended with the data object once specified. If the data object extended parameters are irrelevant to a particular simulation, then they just can be ignored in the analysis. In the script VIII.5 we described each line in more details.

Listing VIII.5: optimization script

```
db = DomainDBInter(database = 'head_modeling.h5')    #specify the domain db.

#get a domain simulation object to be used in the analysis
sim = db.getSimulation('ADI')

#Get the geometry of a subjects' head
geom = db.getObject('Geometry', query='fname=='SomeName' and gresolution==1')

#extend simulation parameters with the data objects' list of parameters.
sim.setParam('geom', geom)

#get a sensors net that is registered to the geometry
sens = db.getObject('Sensors', query='geometry == %s'% geom.name)
sim.setParam('sensors', sens)

#create a simulation manager object that use a pool of 12 workers.
sm = SimMan(workers=12, clusters=['mist'], sim=sim, useDB=False, db=db)

#get a measured data set that is registered to the sens net
measured = db.getObject('CInjection', query='sensors==%s' % sens.name)

#provide an objective function to be optimized
ofun = CondObjFunc(measured)
simout = ['potentials']

varibles =     ['skull',       'brain',     'scalp',     'csf'    ]
lower_bounds = {'skull':0.001, 'brain':.05, 'scalp':.05, 'csf':1.2}
upper_bounds = {'skull':0.1,   'brain':1.0, 'scalp': 1.0,'csf':2.2}
prec         = {'skull': 4,    'brain':3,   'scalp':3,   'csf':3  }
init_values  = {'skull':0.05,  'brain':0.58,'scalp':.55, 'csf':1.4}

#invoke the optimization module.
op1 = Optimization(method=''SA'', simman=sm, vnames = variables, vlb=lower_bounds,
                   vub = upper_bounds, vinit = init_values, vprec = prec,
                   sim_outputs = simout, objfunc = ofun, custom = True)

op1.go()            #start executing of the optimization methods
r1 = opt1.results() #get the results when done
```

The design of ODESSI allows the user to experiment with different objective functions without modifying the simulation code or the environment. The objective function is a user supplied Python callable that provides the objective function to be optimized. The optimization method calls the objective function and passes a dictionary of the names and values of the simulation outputs. To use other data in computing the objective function, the objective function can be implemented as a Python class that implements the built in methods __init__ and __call__. The

Listing VIII.6: objective function for conductivity modeling

```
class CondObjFunc:
    def __init__(self, data):
        self.M = data

    def __call__(self, simout):
        #compute the objective value
        D = simout['potentials'])
        s = sqrt(sum((D[:,1] - self.M[:,1])**2)/(len(D)))
        return s
```

__init__ is where the user provides data that can be used in computing the objective function. The __call__ method is called when an object of the class is called as a function. In this example, the provided objective function is shown in Listing VIII.5. In this case the __init__ method takes the measured data object as an argument in the initialization. Then when it is called, it computes the difference between the measured data set and the computed data set.

8.2.4 ADI Forward Solver Parameter Tuning

Our forward model is based on solving the time-dependent Poisson equation and considering the steady state solution as the static solution. The convergence of the forward solver depends, on two parameters. The *time step*, which controls the speed of reaching the steady state, and the convergence *tolerance* which specifies the level of accuracy. We used ODESSI to tune these parameters by performing a parametric study for different current injection pairs and different sets of conductivities. Figure 8.6 shows a sample from this study. As the tolerance

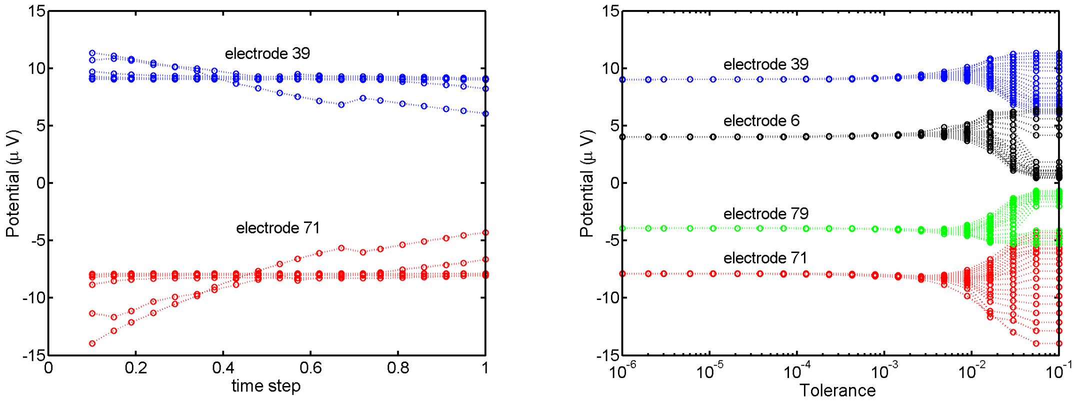


FIGURE 8.6: Tuning the forward-solver convergence parameters.

increases the solution fails to converge. For very small time step, the solver terminates prematurely.

8.2.5 Geometry Resolution Error

The geometry of the head tissues is obtained from imaging such as MRI or CT scans. Geometry obtained from high resolution (1*mm*) MRI captures more details about the head tissues, such as wholes in the skull. However, the computational time is significant. We use a high resolution image to construct lower resolution geometry by eliminating every other plane from the high resolution image. Then we used ODESSI to evaluate the error caused by this approximation. RDM and MAG metrics are used to compare between the solutions obtained using the two geometries. Our results show that the average RDM is about .8 and the average MAG is about .1. Therefore, the 2*mm* geometry can be used for visualization and testing. However, we have to use the high 1*mm* resolution to

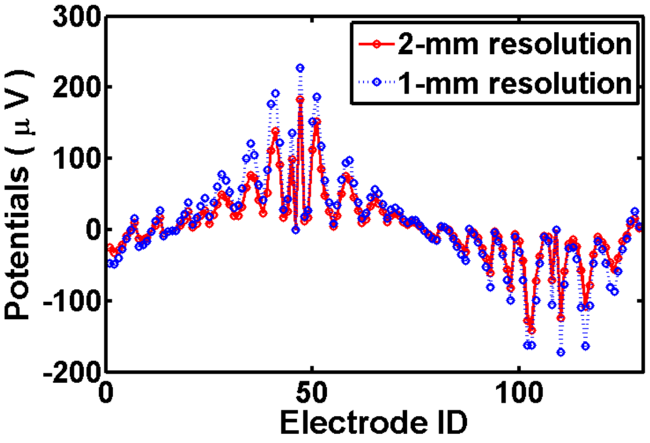


FIGURE 8.7: The potentials at the electrodes using 1*mm* resolution geometry vs. 2*mm* resolution.

obtain accurate conductivity reconstruction. Here ODESSI allows us to experiment with the metrics for comparison. Figure 8.7 shows the potential at the electrodes using 1*mm* and 2*mm* geometry resolutions.

8.3 Summary and Conclusions

In this chapter we used ODESSI to conduct several scientific investigations in two different domains, the human neuroscience domain and the computational chemistry domain. For the computational chemistry domain we used ODESSI to extract the model parameters and to conduct several parametric studies to understand the precision of the model output. In the human neuroscience domain we used ODESSI in tuning the model convergence parameters, extracting the model parameters, studying the effect of the geometry resolution, studying the model output sensitivity to several input parameters, and managing the domain data.

The investigations in this chapter demonstrate how easy it is to apply ODESSI to arbitrary simulation and conduct various kinds of investigations leveraging HPC. Also, they show how ODESSI integrates the three aspects of scientific investigations: the execution of the model, the management of domain data, and the application of scientific methods. The provided investigation scripts realize the integration of these aspects in conducting scientific investigations. This design factors out the computational parts of the scientific investigation which allows scientist to focus on the science aspect of computational science.

CHAPTER IX

CONCLUSION

The main contributions of this dissertation are in two areas.

First, the thesis research resulted in new methods for the computational modeling of human head electromagnetics. The quality and accuracy of forward modeling for simulating electrical fields in the human head volume was improved by the inclusion of skull imhomogeneities and other skull variations. The approach was based on skull parcellation and structural properties obtained from experimental studies. The performance of forward modeling was improved through multi-core (shared memory and accelerator-based) parallelization. Further, the research resulted in an innovative approach for the bounded EIT (bEIT) inverse problem to estimate head tissue conductivities. Parallel search methods were used to improve performance and enable verification and validation of head modeling solutions.

Second, the thesis research contributed to field of scientific computational environments. An architecture was designed (functional and system) for an environment to support simulation-based scientific investigations founded on a framework model that abstracts common scientific methods and provide standard components for problem solving. An approach to capture common scientific methods in a general form was specified for use in scientific investigations, and the approach was realized for a particular set of scientific methods captured in a scientific methods library. A programming model for scientific investigations that provides an abstract interface to scientific methods based on method parameterization was designed and implemented using a scripting language system. A simulation optimization model was created that decides what simulations to conduct based on the scientific method request and the current state of the investigation results. A scientific investigation management system that maintains the evolving record of a scientific study was specified.Lastly, the environment and techniques were evaluated in two application domains: human neuroscience and computational chemistry.

The thesis research pursued principles of separation, abstraction, and integration in three aspects of simulation-based scientific investigations: simulation execution, domain data and investigations management, and the application of scientific methods. Factoring out the application of scientific methods allows the capture of common scientific methods in a general purpose software library which can be applied cross simulation concerns and can be extended independent of the domain data or the execution model. Factoring out the simulation execution allows the environment to leverage available computing resources to meet simulation demands, independent of the data model or the scientific methods. It also allows the reuse and sharing of previous simulation results. Factoring out the domain data allows the application of the environment across multiple domains while enabling domain specificity of scientific procedures. Through the use of scripting approaches to program scientific investigations, the research further demonstrated how the execution, specification, and logic of a scientific method can be explicitly separated. This allowed for customization of scientific methods from generic specification, for instantiation of domain data objects from data schemas, and for the defininition and reuse of simulation designs.

The thesis delivered a working prototype of a framework for simulation-based science in the ODESSI environemt. The principles and design approach were reflected in the evaluation of ODESSI for computational chemistry and human neuroscience domains. The investigations studied included optimization, sensitivity analysis, and parametric analysis, required intensive computation on distributed resources, and delivered significant results in each field.

BIBLIOGRAPHY

[1] Scientific method. `http://en.wikipedia.org/wiki/Scientific_method`.

[2] D. Abramson, A. Lewis, and T. Peachey. Nimrod/o: A tool for automatic design optimization. In *The 4th International Conference on Algorithms & Architectures for Parallel Processing (ICA3PP 2000)*, Hong Kong, China, 2000.

[3] D. Abramson, A. Lewis, T. Peachey, and C. Fletcher. An automatic design optimization tool and its application to computational fluid dynamics. In *Supercomputing '01: Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM)*, pages 25–25, New York, NY, USA, 2001. ACM.

[4] V. Abrashin, A. Egorov, and N. Zhadaeva. On the convergence rate of additive iterative methods. *Differential Equations*, 37:867–879, 2001.

[5] C. Acar and N. G. Gencer. Forward problem solution of esi using fem and bem with quadratic isoparametric elements. In *the First Joint BMESEMBS Conference*, Atlanta, 1999. Omnipress.

[6] E. Akarsu, G. Fox, W. Furmanski, and T. Haupt. Webflow: high-level programming environment and visual authoring toolkit for high performance distributed computing. In *Supercomputing '98: Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM)*, pages 1–7, Washington, DC, USA, 1998. IEEE Computer Society.

[7] R. Akers, E. Kant, C. Randall, S. Steinberg, and R. Young. Scinapse: A problem-solving environment for partial differential equations. *Comp. Sci. and Eng.,*, 4(3), 1997.

[8] M. Akhtari, H. Bryant, D. Emin, W. Merrifield, A. Mamelak, E. Flynn, J. Shih, M. Mandelkern, A. Matlachov, D. Ranken, E. Best, M. DiMauro, R. Lee, and W. Sutherling. A model for frequency dependence of conductivities of the live human skull. *Brain Topography*, 16(1), 2003.

[9] M. Akhtari, H. Bryant, A. Mamelak, E. Flynn, L. Heller, J. Shih, M. Mandelkern, A. Matlachov, D. Ranken, E. Best, M. DiMauro, R. Lee, and W. Sutherling. Conductivities of three-layer live human skull. *Brain Topography*, 14(3):151–167, 2002.

[10] J. Armstrong. *Programming Erlang: Software for a Concurrent World.* Pragmatic Bookshelf, 2007.

[11] J. Ary, S. Klein, and D. Fender. Location of sources of evoked scalp potentials: Corrections for skull and scalp thicknesses. *IEEE Transactions on Biomedical Engineering*, 28(6):447–452, 1981.

[12] K. Awada, D. Jackson, J. Williams, D. Wilton, S. Baumann, and A. Papanicolau. Computational aspects of finite element modeling in eeg source localization. *IEEE Trans. Biomed. Eng.*, 44:736–752, 1997.

[13] A. P. Bagshaw, A. D. Liston, R. H. Bayford, A. Tizzard, A. P. Gibson, A. T. Tidswell, M. K. Sparkes, H. Dehghanin, C. D. Binnies, and D. S. Holder. Electrical impedance tomography of human brain function using reconstruction algorithms based on the finite element method. *NeuroImage*, 20(2):752–764, Oct 2003.

[14] S. Baillet and L. Garnero. A bayesian approach to introducing anatomo-functional priors in the eeg/meg inverse problem. *IEEE Transactions on Biomedical Engineering*, 44(5):374–385, 1997.

[15] S. Baillet, J. Mosher, and L. R.M. Electromagnetic brain mapping. *IEEE Signal Processing Magazine*, 18(6):14–30, 2001.

[16] M. Banich. *Cognitive Neuroscience and Neuropsychology.* Houghton Mifflin,, New York, 2004.

[17] A. Barnard, I. Duck, and M. Lynn. The application of electromagnetic theory to electrocardiography: I. derivation of the integral equations. *Biophysics Journal 1967*, 7((5)):443–462, 1967.

[18] A. Barnard, I. Duck, M. Lynn, and W. Timlake. The application of electromagnetic theory to electrocardiography. *Biophys. J.*, 7:433–462, 1968.

[19] P. Basser, J. Mattiello, and D. LeBihan. Mr diffusion tensor spectroscopy and imaging. *Biophysical Journal*, 66:259–267, 1994.

[20] S. Baumann, D. Wozny, S. Kelly, and F. Meno. The electrical conductivity of human cerebrospinal fluid at body temperature. *IEEE Trans. Biomed. Eng.*, 44(3):220–223, 1997.

[21] C. G. Benar and J. Gotman. Modeling of post-surgical brain and skull defects in the eeg inverse problem with the boundary element method. *Clin. Neurophysiol*, 113:48–56, 2002.

[22] P. Berg and M. Scherg. A fast method for forward computation of multiple-shell spherical head models. *Electroencephalography and Clinical Neurophysiology*, 90:58–64, 1994.

[23] O. Bertrand, M. Thevenet, and F. Perrin. 3-d finite element method in brain electrical activity studies. Technical report, TKK-F-A689-HUT, 1991.

[24] The biomedical informatics research network (birn). `http://www.birncommunity.org/`.

[25] A. Borsic. *Regularization methods for imaging from electrical measurement.* PhD thesis, Oxford Brookes University, 2002.

[26] B. H. Brown, D. Barber, and A. D. Seagar. Applied potential tomography: possible clinical applications. *Clin. Phys. Physiol Meas*, 6(2):109–121, May 1985.

[27] H. Buchner, G. Knoll, M. Fuchs, A. Rienacker, R. Beckmann, M. Wagner, J. Silny, and J. Pesch. Inverse localization of electric dipole current sources in finite elements models of the human head. *Electroenceph Clin. Neurophysiol*, 102:267–278, 1997.

[28] H. Casanova, T. Bartol, J. Stiles, and F. Berman. Distributing mcell simulations on the grid. *Int. J. High Perform. Comput. Appl.*, 15(3):243–257, 2001.

[29] H. Casanova, G. Obertelli, F. Berman, and R. Wolski. The AppLeS parameter sweep template: User-level middleware for the grid. pages 75–76, 2000.

[30] Common data format (cdf). `http://cdf.gsfc.nasa.gov/`.

[31] D. B. Cedrés and E. Hernández. Parameter sweeping methodology for integration in a workflow specification framework. In *ICCSA (1)*, pages 360–371, 2007.

[32] R. chapman, R. Ilmoniemi, B. S., and R. GL. Selective localization of alpha brain activity with neuromagnetic measurements. *Electroencephalogr. Clin. Neurophysiol.*, 58:569–572, 1984.

[33] N. Chauveau, X. Franceries, B. Doyon, B. Rigaud, J. Morucci, and P. Celsis. Effects of skull thickness, anisotropy, and inhomogeneity on forward eeg/erp computations using a spherical three-dimensional resistor mesh model. *Human Brain Mapping*, 21(2):86–97, 2003.

[34] N. Chauveau, X. Franceries, B. Doyon, B. Rigaud, J. Morucci, and P. Celsis. Effects of skull thickness, anisotropy, and inhomogeneity on forward eeg/erp computations using a spherical three-dimensional resistor mesh model. *Hum. Brain. Mapp.*, 21:86–97, 2004.

[35] S. Cherry and M. Phelps. Imaging brain function with positron emission tomography. *Brain Mapping: The Methods*, 1996.

[36] D. Churches, G. Gombas, A. Harrison, J. Maassen, C. Robinson, M. Shields, I. Taylor, and I. Wang. Programming Scientific and Distributed Workflow with Triana Services. *Concurrency and Computation: Practice and Experience (Special Issue: Workflow in Grid Systems)*, 18(10):1021–1037, 2006.

[37] Chemical markup language (cml). `http://www.ch.ic.ac.uk/rzepa/cml/`.

[38] D. Cohen, B. Cuffin, K. Yunokuchi, R. Maniewski, C. Purcell, G. Cosgrove, J. Ives, J. Kennedy, and D. Schomer. Meg versus eeg localization test using implanted sources in the human brain. *Ann. Neurol.*, 28:811–817, 1990.

[39] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the &ldquo;simulated annealing&rdquo; algorithm. *ACM Trans. Math. Softw.*, 13(3):262–280, September 1987.

[40] A. Crouzeix, B. Yvert, O. Bertrand, and J. Pernier. An evaluation of dipole reconstruction accuracy with spherical and realistic head models in meg. *Clin. Neurophysiol.*, 110(12):2176–2188, 1999.

[41] B. Cuffin. Effects of head shape on eegs and megs. *IEEE Trans Biomed. Eng.*, 37:699–705, 1990.

[42] B. Cuffin and D. Cohen. Magnetic fields of a dipole in special volume conductor shapes. *IEEE Trans Biomed. Eng.*, 24:371–381, 1977.

[43] B. Cuffin, D. Cohen, K. Yunokuchi, R. Maniewski, G. Cosgrove, J. Ives, J. Kennedy, and D. Schomer. Tests of eeg localization accuracy using implanted sources in the human brain. *Annals of neurology*, 29:132–138, 1991.

[44] B. N. Cuffin. Effects of local variations in skull and scalp thickness on eegs and megs. *IEEE Trans. Biomed. Eng.*, 40:42–48, 1993.

[45] N. Cuffin. Eeg localization accuracy improvements using realistically shaped head models. *IEEE Trans Biomed. Eng.*, 43, 1996.

[46] J. Cuny and et al. Building domain–specific environments for computational science:a case study in seismic tomg. *Int. Journal of Super. App. and High Speed Computing*, 11(3), 1997.

[47] A. D., J. Giddy, and L. Kotler. High performance parametric modeling with nimrod/G: Killer application for the global grid?, 2000.

[48] F. Darvas, D. Pantazis, E. Pantazis, Kucukaltun-Yildirim, and L. R.M. Mapping human brain function with meg and eeg: methods and validation. *Mathematics in Brain Imaging*, 23:289–299, 2004.

[49] J. De Munck. The estimation of time varying dipoles on the basis of evoked potentials. *Electroencephalography and Clinical Neurophysiology*, 77:156, 1990.

[50] J. de Munck and M. Peters. A fast method to compute the potential in the multiphere model. *IEEE Transactions on Biomedical Engineering*, 40(11):1166–1174, 1993.

[51] J. de Munck, B. van Dijk, and H. Spekreijse. Mathematical dipoles are adequate to describe realistic generators of human brain activity. *IEEE Trans. Biomed. Eng.*, 35(11):960–966, November 1988.

[52] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, and M. Livny. Pegasus: Mapping scientific workflows onto the grid. *Lecture Notes in Computer Science : Grid Computing*, pages 11–20, 2004.

[53] L. Ding and B. Lai. Low resolution brain electromagnetic tomography in a realistic geometry head model: a simulation study. *Phys. Med. Biol.*, 50:45–56., 2005.

[54] D. Dobbs. Fact or phrenology? *Scientific American Mind*, April 2005. `http://www.scientificamerican.com/article.cfm?id=fact-or-phrenology`.

[55] J. Douglas and H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Trans. Am. Math. Soc.*, 82:421–439, 1956.

[56] Extensible computational chemistry environment. `http://ecce.emsl.pnl.gov/`.

[57] Electrical geodesics, inc. `http://www.mcs.anl.gov/petsc/petsc-as/`.

[58] Enthought. `http://www.enthought.com/`.

[59] J. Ermer, J. Mosher, S. Baillet, and R. Leahy. Rapidly recomputable eeg forward models for realistic headshapes. *Phys. Med. Biol.*, 46(4):1265–1281, 2001.

[60] Y. Eshel, S. Witman, M. Rosenfeld, and S. Abboud. Correlation between skull thickness asymmetry and scalp potential estimated by a numerical model of the head. *IEEE Trans. Biomed. Eng.*, 42:242–249, Mar 1995.

[61] T. Ferree, K. Eriksen, and D. Tucker. Regional head tissue conductivity estimation for improved eeg analysis. *IEEE Trans. Biomed. Eng.*, 47(12):1584–92, 2000.

[62] Definition of the flexible image transport system (fits). `http://fits.gsfc.nasa.gov/standard30/fits_standard30.pdf`.

[63] L. Flemming, Y. Wang, A. Caprihan, M. Eiselt, J. Haueisen, and Y. Okada. Evaluation of the distortion of eeg signals caused by a hole in the skull mimicking the fontanel in the skull of human neonates. *Clin. Neurophysiol*, 116(5):1141–1152, 1984.

[64] E. Frank. Electric potential produced by two point current sources in a homogeneous conduction sphere. *Applied Physics*, 23(11):1225–1228, 1952.

[65] W. J. Freemana, S. P. Ahlforsb, and V. Menon. Combining fmri with eeg and meg in order to relate patterns of brain activity to cognition. *Journal of Psychophysiology*, 73:43–52, 2009.

[66] I. Frerichs, P. Braun, T. Dudykevych, G. Hahn, D. Genee, and G. Hellige. Distribution of ventilation in young and elderly adults determined by electrica impedance tomography. *Respir Physiol Neurobiol*, 143:63–75, 2004.

[67] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke. Condor-g: A computation management agent for multi-institutional grids. *hpdc*, 00:0055, 2001.

[68] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke. Condor-g: A computation management agent for multi-institutional grids. *hpdc*, 00:0055, 2001.

[69] M. Fuchs, M. Wagner, and J. Kastner. Boundary element method volume conductor models for eeg source reconstruction. *Clin. Neurophys.*, 112:1400–1407, 2001.

[70] C. Gabriel, S. Gabriel, and E. Corthout. The dielectric properties of biological tissues: Part i. literature survey. *Phys. Med. Biol.*, 41:2231–2249, 1996.

[71] S. Gabriel, R. Lau, and C. Gabriel. The dielectric properties of biological tissues: Part ii. measurements in the frequency range 10 h_z to 20 gh_z. *Phys. Med. Biol.*, 41:2251–2269, 1996.

[72] S. Gabriel, R. Lau, and C. Gabriel. The dielectric properties of biological tissues: Part iii. parametric models for the dielectric spectrum of tissues. *Phys. Med. Biol.*, 41:2271–2293, 1996.

[73] E. Gallopoulos, E. Houstis, and J. R. Rice. Computer as thinker/doer: Problem-solving environments for computational science. *IEEE Comput. Sci. Eng.*, 1(2):11–23, 1994.

[74] M. Gazzaniga and T. Heatherton. *Psychological Science: Mind, Brain, and Behavior.* W. W. Norton & Company, New York, 2003.

[75] G. Geist, J. Kohl, and P. Papadopoulos. Cumulvs: Providing fault-tolerance, visualization and steering of parallel applications. In *Environment and Tools for Parallel Scientifc Computing Workshop (Domaine de Faverges-de-la-Tour)*, Lyon, France, 1996. IEEE Press.

[76] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers. Examining the challenges of scientific workflows. *Computer*, 40(12):24–32, 2007.

[77] P. Gloor. Neuronal generators and the problem of localization in electroencephalography: Application of volume conductor theory to electroencephalography. *Clin. Neurophysiol.*, 2:327–354, 1985.

[78] Geography markup language (gml). `http://en.wikipedia.org/wiki/Geography_Markup_Language`.

[79] S. Goncalves, J. de Munck, J. Verbunt, R. Heethaar, and L. da Silva F.H. In vivo measurement of the brain and skull resistivities using an eit-based method and the combined analysis of sef/sep data. *IEEE Transactions on Biomedical Engineering*, 50(9):1124–8, Sept 2003.

[80] S. Gonçalves, J. de Munck, J. Verbunt, F. Bijma, R. M. Heethaar, and F. da Silva. In vivo measurement of the brain and skull resistivities using an eit-based method and realistic models for the head. *IEEE Transactions on Biomedical Engineering*, 50(6):754–767, June 2003.

[81] J. Gotman, E. Kobayashi, A. Bagshaw, C. Benar, and F. Dubeau. Combining eeg and fmri: a multimodal tool for epilepsy research. *Magn Reson Imaging*, 23(6):906–20, 2006.

[82] R. Grech, T. Cassar, J. Muscat, K. P. Camilleri, S. G. Fabri1, M. Zervakis, P. Xanthopoulos, V. Sakkalis, and B. Vanrumste. Review on solving the inverse problem in eeg source analysis. *NeuroEngineering and Rehabilitation*, 5(25):1–33, November 2008. `http://www.jneuroengrehab.com/content/5/1/25`.

[83] GridLab. `http://www.gridlab.org/`.

[84] D. Gullmar, J. Haueisen, M. Eiselt, F. Giessler, L. Flemming, A. Anwander, T. Knosche, C. Wolters, M. Dumpelmann, D. Tuch, and J. Reichenbach. Influence of anisotropic conductivity on eeg source reconstruction: Investigations in a rabbit model. *IEEE Trans. Biomed. Eng.*, 53:1841–1850, 2006.

[85] H. Hallez, P. Van Hese, B. Vanrumste, P. Boon, Y. D'Asseler, I. Lemahieu, and R. Van de Walle. Dipole localization errors due to not incorporating compartments with anisotropic conductivities: Simulation study in a spherical head model. *International Journal of Bioelectromagnetism*, 7:134–137, 2005.

[86] H. Hallez, B. Vanrumste, R. Grech, J. Muscat, W. De Clercq, A. Vergult, Y. D'Asseler, K. Camilleri, S. Fabri, S. Van Huffel, and I. Lemahieu. Review on solving the forward problem in eeg source analysis. *Journal of NeuroEngineering and Rehabilitation*, 4(46), 2007.

[87] H. Hallez, B. Vanrumste, P. Van Hese, Y. D'Asseler, I. Lemahieu, and R. Van de Walle. A finite difference method with reciprocity used to incorporate anisotropy in electroencephalogram dipole source localization. *Physics in Medicine and Biology*, 50:3787–3806, 2005.

[88] M. Hämäläinen and J. Sarvas. Realistic conductivity geometry model of the human head for interpretation of neuromagnetic data. *IEEE Trans Biomed Eng*, 36:165–171, Feb 1989.

[89] M. S. Hämäläinen, R. Hari, R. J. Ilmoniemi, J. Knuutila, and O. V. Lounasmaa. Magnetoencephalography theory, instrumentation, and applications to noninvasive studies of the working human brain. *Rev. Mod. Phys.*, 65(12):413–497, 1993.

[90] J. Haueisen, C. Ramon, M. Eiselt, H. Brauer, and H. Nowak. Influence of tissue resistivities on neuromagnetic fields and electric potentials studied with a finite element model of the head. *IEEE Trans., Biomed. Eng.*, 44(8), 1997.

[91] J. Haueisen, D. Tuch, C. Ramon, P. Schimpf, V. Wedeen, J. George, and J. Belliveau. The influence of brain tissue anisotropy on human eeg and meg. *Neuroimage*, 15:159–166, 2002.

[92] J. Hauesien, A. Böttner, H. Nowak, H. Brauer, and C. Weiller. The influence of conductivity changes in boundary element compartments on the forward and inverse problem in electroencephalography and magnetoencephalography. *Biomed. Tech.*, 44(6):150–157, 1999.

[93] An introduction to hdf5. `http://www.hdfgroup.org/HDF5/doc/H5.intro.html`.

[94] D. J. Heeger and D. Ress. What does fmri tell us about neuronal activity? *Nature Reviews Neuroscience*, 3:142–151, 2002. `http://www.nature.com/nrn/journal/v3/n2/full/nrn730.html`.

[95] M. Hobbs. Candygram. `http://candygram.sourceforge.net/`.

[96] R. Hoekema, W. G.H., L. F.S.S., van Veelen C.W.M., van Rijen P.C., H. G.J.M., A. J., and van Huffelen A.C. Measurement of the conductivity of skull, temporarily removed during epilepsy surgery. *Brain Topography*, 16(1):29–38, 2003.

[97] R. Hoekema, K. Venner, J. Struijk, and J. Holsheimer. Multigrid solution of the potential field in modeling electrical nerve stimulation. *Computers and Biomedical Research*, 31:348–362, 1998.

[98] D. S. Holder. *Electrical Impedance Tomography: Methods, History and Applications*. Taylor & Francis, 2004.

[99] J. Hong, D. H. Jeong, C. D. Shaw, W. Ribarsky, M. Borodovsky, and C. G. Song. Gvis: A scalable visualization framework for genomic data. In K. Brodlie, D. J. Duke, , and K. I. Joy, editors, *EuroVis*, pages 191–198. Eurographics Association, 2005.

[100] R. Hopfengartner, F. Kerling, V. Bauer, and S. H. An efficient, robust and fast method for the offline detection of epileptic seizures in long-term scalp eeg recordings. *Clin Neurophysiol*, 118(11):2332–43, Nov 2007.

[101] E. houstis, J. Rice, E. Gallopoulos, and R. Bramley. *Enabling technologies for computational science*. Kluwer academic publishers, 2000.

[102] G. Huiskamp, J. Maintz, G. Wieneke, M. Viergever, and A. van Huffelen. The influence of the use of realistic head geometry in the dipole localization of interictal spike activity in mtle patients. *Biomedizinische Technik.*, 42:84–87, 1997.

[103] G. Huiskamp, M. Vroeijenstijn, R. van Dijk, G. Wieneke, and A. Huffelen. The need for correct realistic geometry in the inverse eeg problem. *IEEE Trans. Biomed. Eng.*, 46:1281–1287, Nov 1999.

[104] Ibm cell broadband engine software development kit. `http://www.ibm.com/developerworks/power/cell/`.

[105] Neurons in a column. `http://domino.watson.ibm.com/comm/pr.nsf/pages/rsc.bluegene_cognitive.html`.

[106] L. Ingber. Simulated annealing: Practice versus theory. *Mathl. Comput. Modelling*, 18(11):29–57, 1993.

[107] J. Jin. *The Finite Element Method in Electromagnetics.* John Wiley & Sons, New York, 1993.

[108] L. Jing. Effects of holes on eeg forward solutions using a realistic geometry head model. *Journal of neural engineering*, 4(3):197–204, 2007.

[109] H. J.S., N. R.R., and A. F.C. Simulated parallel annealing within a neighborhood for optimization of biomechanical systems. *Biomech*, 2005.

[110] C. Junwei, S. Jarvis, S. Saini, and G. Nudd. Gridflow: Workflow management for grid computing. *ccgrid*, 0:198, 2003.

[111] K. K. and T. K. Time-homogeneous parallel annealing algorithm. In *The 13th IMACS World Congress of Computation and Applied Mathematics*, 1991.

[112] P. Kacsuk, Z. Farkas, G. Hermann, and T. Kiss. Workflow-level parameter study support for production grids. In O. Gervasi and M. Gavrilova, editors, *Computational science and its application - ICCSA 2007. International conference. Part III. Kuala Lumpur, 2007. (Lecture notes in computer science 4707)*, pages 872–885. LNCS, 2007.

[113] J. Kaipio, V. Kolehmainen, M. Vauhkonen, and E. Somersalo. Inverse problems with structural prior information. *Inverse problems*, 15:713–729, Nov 1998.

[114] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[115] J. Kosterich, K. Foster, and S. Pollack. Dielectric properties of fluid saturated bone: The effect of variation in conductivity of immersion fluid. *IEEE Trans. Biomed. Engr.*, 31:369–375, 1984.

[116] Y. Lai, W. van Drongelenc, L. Dinga, K. Hecoxc, V. Towlec, D. Frimc, and B. Hea. Estimation of in vivo human brain-to-skull conductivity ratio from simultaneous extra- and intra-cranial electrical potential recordings. *Clinical Neurophysiology*, 116(2):456–465, 2005.

[117] J. Latikka, T. Kuurne, and H. Eskola. Conductivity of living intracranial tissues. *Phys. Med. Biol.*, 46(6):1611–6, 2001.

[118] S. Law. Thickness and resistivity variations over the upper surface of the human skull. *Brain Topography*, 6:99–109, 1993.

[119] L. Lemieux, A. McBride, and J. Hand. Calculation of electrical potentials on the surface of a realistic head model by finite differences. *Phys. Med. Biol.*, 41:1079–1091, 1996.

[120] W. R. B. Lionheart. Eit reconstruction algorithms: pitfalls, challenges and recent developments. *Physiol. Meas.*, 25:125–142, 2004.

[121] N. K. Logothetis. What we can do and what we cannot do with fmri. *Nature*, 453:869–878, 12,June 2008.

[122] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the kepler system: Research articles. *Concurr. Comput. : Pract. Exper.*, 18(10):1039–1065, 2006.

[123] R. MacLeod and et al. Scirun/biopse: Integrated problem solving environment for bioelectric field problems and visualization. In *IEEE Int. Symp. on Biom. Img.*, volume 1, pages 640–643, 2004.

[124] Microarray and gene expression markup language (mage–ml). `http://xml.coverpages.org/mageML.html`.

[125] G. Marin, C. Guerin, S. Baillet, L. Garnero, and G. Meunier. Influence of skull anisotropy for the forward and inverse problem in eeg: simulation studies using fem on realistic head models. *Hum. Brain. Mapp.*, 6:250–269, 1998.

[126] J. Meijs, O. Weier, M. Peters, and A. van Oosterom. On the numerical accuracy of the boundary element method. *Biomed. Eng.*, 36:1038–1049, Oct 1989.

[127] J. Mejis, M. Peters, and A. Oosterom. Computation of megs and eegs using a realistically shaped multicompartment model of the head. *Med. Biol. Eng.*, 23:36–37, 1985.

[128] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[129] M. Miki, T. Hiroyasu, M. Kasai, K. Ono, and T. Jitta. Temperature parallel simulated annealing with adaptive neighborhood for continuous optimization problem. In *Second international workshop on Intelligent systems design and application*, pages 149–154, Atlanta, GA, USA, 2002. Dynamic Publishers, Inc.

[130] Molecular dynamics [markup] language (modl). `http://xml.coverpages.org/modl.html`.

[131] M. Mohr and B. Vanrumste. Comparing iterative solvers for linear systems associated with the finite difference discretisation of the forward problem in electro-encephalographic source analysis. *Med. Biol. Eng. Comput.*, 41:75–84, 2003.

[132] Z. Molnar and I. Szeberenyi. Saleve: Simple web-services based environment for parameter study applications. In *GRID 05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pages 292–295, Washington, DC, USA, 2005. IEEE Computer Society.

[133] J. Mosher and R. Leahy. Source localization using recursively applied and projected (rap) music. *IEEE Trans. Signal. Process.*, 47(2):332–340, 1999.

[134] Numerical data markup language (ndml). `http://xml.coverpages.org/materials.html`.

[135] The network for earthquake engineering simulation (nees). `https://www.nees.org/`.

[136] L. Neilson, M. Kovalyov, and Z. Koles. A computationally efficient method for accurately solving the eeg forward problem in a finely discretized head model. *Clin Neurophysiol*, 116(10):2302–2314, 2005.

[137] Network common data format (netcdf). `http://www.unidata.ucar.edu/software/netcdf/`.

[138] P. Nicholson. Specific impedance of cerebral white matter. *Experimental Neurology*, 13:386–401, 1965.

[139] Numerical python. adds a fast, compact, multidimensional array language facility to python. `http://www.pfdubois.com/numpy/`.

[140] P. Nunez. *Electric Fields of the Brain: The neurophysics of the EEG*. Oxford University Press, 2006.

[141] P. Nunez and R. Silberstein. On the relationship of synaptic activity to macroscopic measurements: Does co-registration of eeg with fmri make sense? *Brain Topogr*, 13:79–96, 2000.

[142] Nwchem homepage. `http://www.emsl.pnl.gov/capabilities/computing/nwchem/`.

[143] A. I. of Aeronautics and A. Staf. *AIAA Guide for the Verification and Validation of Computational Fluid Dynamics Simulations*. American Institute of Aeronautics & Astronautics, 1998.

[144] S. Ogawa, D. Tank, R. Menon, J. Ellermann, S. Kim, H. Merkle, and K. Ugurbil. Intrinsic signal changes accompanying sensory stimulation: functional brain mapping with magnetic resonance imaging. *Natl. Acad. Sci.*, 89(13):5951–5955, 1992.

[145] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, T. Carver, M. Greenwood, K. Glover, M. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, June 2004.

[146] Y. Okada, A. Lahteenmaki, and C. Xu. Experimental analysis of distortion of magnetoencephalography signals by skull. *Clin. Neurophysiol.*, 110(2):230–238, 1999.

[147] J. Ollikainen, M. Vauhkonen, P. Karjalainen, and J. Kaipio. Effects of local skull inhomogeneities on eeg source estimation. *Med. Eng. Phys.*, 21(3):143–154, 1999.

[148] T. Oostendorp, J. Delbeke, and D. Stegeman. The conductivity of the human skull: Results of in vivo and in vitro measurements. *IEEE Transactions on Biomedical Engineering*, 47(11):1487–1492, 2000.

[149] R. Oostenveld and T. Oostendorp. Validating the boundary element method for forward and inverse eeg computations in the presence of a hole in the skull. *Hum. Brain. Mapp.*, 17:179–192., 2002.

[150] R. Pascual-Marqui. Review of methods for solving the eeg inverse problem. *International Journal of Bioelectromagnetism*, 1:75–6, 1999.

[151] A. Petitet, H. Casanova, R. Whaley, J. Dongarra, and Y. Robert. A numerical linear algebra problem-solving environment designer's perspective. *SIAM Annual Meeting,*, 1999.

[152] Petsc: Portable, extensible toolkit for scientific computation,. `http://www.mcs.anl.gov/petsc/petsc-as/`.

[153] R. Plonsey and D. B. Heppner. Considerations of quasi-stationarity in electrophysiological systems. *Bulletin of Mathematical Biology*, 29(4):657–664, December 1967.

[154] R. Pohlmeier, H. Buchner, G. Knoll, A. Rienäcker, R. Beckmann, and J. Pesch. The influence of skull - conductivity misspecification on inverse source localization in realistically shaped finite element head models. *Brain Topogr.*, 9(3):157–162, 1997.

[155] M. Posner and M. Raichel. *Images of Mind.* W.H. Freeman, New York, 1997.

[156] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *The Numerical Recipes in C: The art of Scientific Computing.* Cambridge University Press, New York, 2nd edition, 1992.

[157] T. D. Project. `http://www.cs.sandia.gov/DAKOTA/`.

[158] A. Pursula, J. Nenonen, E. Somersalo, E. Ilmoniemi, and T. Katila. Bioelectromagnetic calculations in anisotropic volume conducters. In *Biomag2000*, pages 659–662, 2000.

[159] Pytables. `http://pytables.sourceforge.net`.

[160] R. K. Rew and G. P. Davis. Netcdf: An interface for scientific data access. *IEEE Computer Graphics and Applications*, 10(4):76–82, 1990.

[161] J. Rice. Computational science and the future of computing research. In *IEEE Computational Science and engineering magazine*, pages 35–41, 1995.

[162] J. R. Rice and R. F. Boisvert. From scientific software libraries to problem-solving environments. *IEEE Computational Science & Engineering*, 3(3):44–53, Fall 1996.

[163] M. E. Riley, R. Buss, R. Campbell, M. Hopkins, P. Miller, A. Moats, and W. Wampler. Verification and validation plan for the codes lsp and icarus (pegasus). Technical report, Sandia National Laboratories, 2002.

[164] P. Roache. *Fundamentals of Computational Fluid Dynamics.* Hermosa Publishers, 1998.

[165] A. Rodriguez-Rivera, B. Van Veen, and R. Wakai. Statistical performance analysis of signal variance–based dipole models for meg/eeg source localization and detection. *IEEE Transactions on Biomedical Engineering*, 50(2):137–149, 2003.

[166] T. Rogers, J. Hocking, U. Noppeney, A. Mechelli, M. Gorno-Tempini, K. Patterson, and C. Price. Anterior temporal cortex and semantic memory: reconciling findings from neuropsychology and functional imaging. *Cogn. Affect Behav. Neurosci.*, 6(3):201–213, 2006.

[167] B. Roth, A. Gorbach, and S. Sato. How well does a three-shell model predict positions of dipoles in a realistically shaped head? *Phys. Med. Biol.*, 87:175–184, 1993.

[168] B. Roth, D. Ko, I. von Albertini-Carletti, D. Scaffidi, and S. S. Dipole localization in patients with epilepsy using the realistic shaped head model. *Electroencephalography and Clinical Neurophysiology*, 102(3):160–166, 1997.

[169] S. Rush and D. Driscoll. current distribution in the brain from surface electrodes. *Anesth. Analg.*, 47(6):717–723, 1968.

[170] S. Rush and D. Driscoll. Eeg electrode sensitivity – an application of reciprocity. *IEEE Trans. Biom. Eng.*, 16(1):15–22, 1969.

[171] S. Rush and D. Driscoll. Eeg electrode sensitivity–an application of reciprocity. *IEEE Trans. Biomed. Eng.*, 16:15–22, 1969.

[172] S. M. Sait and Y. Habib. *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems.* IEEE Computer Society Press, Los Alamitos, CA, USA, 1999.

[173] H. Saleheen and T. Kwong. New finite difference formulations for general inhomogeneous anisotropic bioelectric problems. *IEEE Trans. Biomed. Eng.*, 44:800–9, 1997.

[174] H. Saleheen and T. Kwong. A new three-dimensional finite-difference bidomain formulation for inhomogeneous anisotropic cardiac tissues. *IEEE Trans. Biomed. Eng.*, 45:15–24, 1998.

[175] A. Salman, A. Malony, and M. Sottile. An open domain-extensible environment for simulation-based scientific investigation (odessi). In *International Conference on Computational Science (ICCS 2009)*, pages 23 – 32, 2009.

[176] A. Salman, A. Malony, S. Turovets, and T. D. Use of parallel simulated annealing for computational modeling of human head conductivity. In *International Conference on Computational Science (ICCS 2007)*, pages 86–93, 2007.

[177] A. Salman, A. Malony, S. Turovets, and T. D. On the role of skull parcellation in the computational modeling of human head conductivity. In *Electrical Impedance Tomography Conference (EIT Conference 2008)*, pages 16–18, Dartmouth College, New Hampshire, 2008.

[178] A. Salman, A. Malony, S. Turovets, A. Morris, and D. Tucker. Modeling human head electromagnetics on the cell broadband engine (poster). Workshop on Solving Computational Challenges in Medical Imaging, 2007. Seattle.

[179] A. Salman, S. Turovets, A. Malony, J. Eriksen, and T. D. Computational modeling of human head conductivity. In *International Conference on Computational Science (1)*, pages 631–638, 2005.

[180] A. Salman, S. Turovets, A. Malony, P. Poolman, C. Davey, J. Eriksen, and T. D. Noninvasive conductivity extraction for high-resolution eeg source localization. *Advances in Clinical Neuroscience and Rehabilitation*, 6(1):27–28, March 2006.

[181] A. Salman, S. Turovets, A. Malony, and V. Volkov. Multi-cluster, mixed-mode computational modeling of human head conductivity. In *International Workshop on OpenMP (IWOMP)*, 2005.

[182] A. Saltelli. Senstivity analysis for importance assessment. *Risk Analysis*, 22(3):579–590, June 2002.

[183] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola. *Global Sensitivity Analysis: The Primer.* Wiley, 2008.

[184] Y. Salu, L. Cohen, D. Rose, S. Sato, C. Kufta, and H. M. An improved method for localizing electric brain dipoles. *IEEE Transactions on Biomedical Engineering*, 37(7):699–705, 1990.

[185] San diego supercomputer center. `http://www.sdsc.edu/`.

[186] J. Sarvas. Basic mathematical and electromagnetic concepts of the biomagnetic inverse problem. *Phys. Med. Biol.*, 32:11–22, 1993.

[187] P. Schimpf, C. Ramon, and H. J. Dipole models for the eeg and meg. *IEEE Transactions on Biomedical Engineering*, 49(5):409–418, 2002.

[188] L. Schwer. An overview of the ptc 60/v&v 10: guide for verification and validation in computational solid mechanics: Transmitted by l. e. schwer, chair ptc 60/v&v 10. *Eng. with Comput.*, 23(4):245–252, 2007.

[189] L. E. Schwer and W. L. Oberkampf. Special issue on verification and validation. *Eng. with Comput.*, 23(4):243–244, 2007.

[190] L. E. Schwer and W. L. Oberkampf. Special issue on verification and validation. *Eng. with Comput.*, 23(4), 2007.

[191] Siam. `http://www.siam.org/students/resources/report.php`.

[192] N. Soni, A. Hartov, C. Kogel, S. Poplack, and K. Paulsen. Multi-frequency electrical impedance tomography of the breast: new clinical results. *Physiol. Meas.*, 25:301–314, 2004.

[193] G. P. Steven and R. J. Christopher. Scirun: a scientific programming environment for computational steering. In *Supercomputing '95: Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM)*, page 52, New York, NY, USA, 1995. ACM.

[194] J. Stinstra and M. Peters. The volume conductor may act as a temporal filter on the ecg and eeg. *Medical and Biological Engineering and Computing*, 36(6):711–716, November 1998.

[195] B. Sylvain, G. Line, M. Gildas, and H. Jean-paul. Combined meg and eeg source imaging by minimization of mutual information. *IEEE Trans. Biomed. Eng*, 46:522–534, 1999.

[196] H. Szu and R. Hartley. Fast simulated annealing. *Physics Letters A*, 122:157–162, 1987.

[197] C. Tang, F. You, G. Cheng, D. Gao, F. Fu, and G. Yang. Correlation between structure and resistivity variations of the live human skull. *IEEE Transactions on Biomedical Engineering*, 55(9):2286–2292, Sept 2008.

[198] I. Tanzer and N. Gencer. A new boundary element method formulation of the forward problem solution of electro-magnetic source imaging. In *In Proc. 19 th Ann. Int. Conf. IEEE Eng. Med. Biol. Soc.*, pages 2100–2103, Chicago,USA, 1997.

[199] S. Teukolsky, W. Vetterling, and B. Flannery. *The Numerical Recipes in C: The art of Scientific Computing. 2nd edition.* Cambridge University Press, New York, 1992.

[200] O. Thevenet, B. O., F. Perrin, T. Dumont, and J. Pernier. The finite element method for a realistic head model of electrical brain activities: Preliminary results. *Clin. Neurophys.*, 12, 1991.

[201] Traits ui. `http://code.enthought.com/projects/traits/docs/html/TUIUG/index.html`.

[202] D. Tuch, V. Wedeen, A. Dale, J. George, and J. Belliveau. Conductivity tensor mapping of the human brain using diffusion tensor mri. *Proc. Natl. Acad. Sci.*, 98(20):11697–11701, 2001.

[203] R. Uitert, D. Weinstein, and C. Johnson. Can a spherical model substitute for a realistic head model in forward and inverse meg simulations? In *13th Int. Conf. on Biomagnetism,*, pages 798–800, 2002.

[204] R. Uitert, D. Weinstein, and C. Johnson. Volume currents in forward and inverse magnetoencephalographic simulations using realistic head models. *Ann. Biomed. Eng.*, 31:21–31, 2003.

[205] S. van den Broek, F. Reiders, M. Donderwinkel, and M. Peters. Volume conduction effects in eeg and meg. *Electroencephalogr. Clin. Neurophysiol.*, 106, 1998.

[206] S. van den Broek, H. Zhou, and M. Peters. Computation of neuromagnetic fields using finite-element method and biot-savart law. *Med. Biol. Eng.*, 34:21–26, 1996.

[207] J. D. Van Horna, R. Poldrackb, and P. A. Functional mri at the crossroads. *International Journal of Psychophysiology*, 73(1):3–9, 2009.

[208] B. Vanrumste, G. Van Hoey, R. Van de Walle, M. D́Havé, I. Lemahieu, and P. Boon. The validation of the finite difference method and reciprocity for solving the inverse problem in eeg dipole source analysis. *Brain Topogr.*, 14(2):83–92, 2001.

[209] A. Villringer and B. Chance. Noninvasive optical spectroscopy and imaging of human brain function. *Trends Neurosci*, 20:435–442, 1997.

[210] Virolab. `http://www.virolab.org/`.

[211] K. Whittingstall, G. Stroink, L. Gates, and F. A. Effects of dipole position, orientation and noise on the accuracy of eeg source localization. *Biomedical Engineering Online*, 2(14), 2003.

[212] A. Wibisono, Z. Zhao, A. Belloum, and M. Bubak. A framework for interactive parameter sweep applications. In M. Bubak, G. D. van Albada, J. Dongarra, and P. M. A. Sloot, editors, *ICCS (3)*, volume 5103 of *Lecture Notes in Computer Science*, pages 481–490. Springer, 2008.

[213] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[214] C. Wolters, A. Anwander, X. Tricoche, D. Weinstein, M. Koch, and R. MacLeod. Influence of tissue conductivity anisotropy on eeg/meg field and return current computation in a realistic head model: A simulation and visualization study using high-resolution finite element modeling. *neuroimage*, 10(014), 2005.

[215] C. Wolters, A. Anwander, X. Tricoche, D. Weinstein, M. Koch, and R. MacLeod. Influence of tissue conductivity anisotropy on eeg/meg field and return current computation in a realistic head model: A simulation and visualization study using high-resolution finite element modeling. *NeuroImage*, 3:813–826, 2006.

[216] C. wolters, C. Koestler, J. Hardtlein, L. Grasedyck, and H. W. Numerical mathematics of the subtraction method for the modeling of a current dipole in eeg source reconstruction using finite head models. *SIAM J. Sci. Comput*, 40(2), 2007.

[217] C. Wolters, M. Kuhn, A. Anwander, and S. Reitzinger. A parallel algebraic multigrid solver for finite element method based source localization in the human brain. *Computing and Visualization in Science*, 5:165–177, 2002.

[218] X. Xiang, R. Kennedy, and G. Madey. Verification and validation of agent-based scientific simulation models. In *Agent-Directed Simulation Conference*, pages 47–55, San Diego, CA, April 2005.

[219] extensible scientific interchange language (xsil.

[220] S. Y. *Iterative methods for sparse linear systems 2nd edition.* SIAM, 2003.

[221] Y. Yan, P. Nunez, and R. Hart. Finite-element model of the human head: scalp potentials due to dipole sources. *Medical and Biological Engineering and Computing*, 29(5), 9 1991.

[222] M. Yarrow, K. M. McCann, R. Biswas, and R. F. Van der Wijngaart. An advanced user interface approach for complex parameter study process specification on the information power grid. In *GRID '00: Proceedings of the First IEEE/ACM International Workshop on Grid Computing*, pages 146–157, London, UK, 2000. Springer-Verlag.

[223] S.-M. Yau, E. Grinspun, V. Karamcheti, and D. Zorin. Sim-x: parallel system software for interactive multi-experiment computational studies. *ipdps*, 0:116, 2006.

[224] S.-M. Yau, E. Grinspun, V. Karamcheti, and D. Zorin. Simx meets scirun: A component-based implementation of a computational study system. *ipdps*, 0:322, 2007.

[225] B. Yvert, O. Bertrand, M. Thvenet, J. F. Echallier, and J. Pernier. A systematic evaluation of the spherical model accuracy in eeg dipole localization. *Electroencephalogr. Clin. Neurophysiol.*, 102:452–459, 1997.

[226] K. Zhang, K. Damevski, V. Venkatachalapathy, and S. Parker. Scirun2: A cca framework for high performance computing. In *the 9th International Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS 2004)*, Santa Fe, NM, 2004. IEEE Press.

[227] Y. Zhang, W. van Drongelen, and B. He. Estimation of in vivo brain-to-skull conductivity ratio in humans. *Appl Phys Lett.*, 89(22), 2006.