Numerical Solution of 3-D Electromagnetic Problems in Exploration Geophysics and its Implementation on Massively Parallel Computers

Jelena Koldan

Department of Computer Architecture

Universitat Politècnica de Catalunya

Thesis submitted for the degree of

Doctor of Philosophy in Computer Architecture

September, 2013

Supervisor: José María Cela

Abstract

The growing significance, technical development and employment of electromagnetic methods in exploration geophysics have led to the increasing need for reliable and fast techniques of interpretation of 3-D electromagnetic data sets acquired in complex geological environments. The first and most important step to creating an inversion method is the development of a solver for the forward problem. In order to create an efficient, reliable and practical 3-D electromagnetic inversion, it is necessary to have a 3-D electromagnetic modelling code that is highly accurate, robust and very fast. This thesis focuses precisely on this crucial and very demanding step to building a 3-D electromagnetic interpretation method.

The thesis presents as its main contribution a highly accurate, robust, very fast and extremely scalable numerical method for 3-D electromagnetic modelling in geophysics that is based on finite elements and designed to run on massively parallel computing platforms. Thanks to the fact that the finite-element approach supports completely unstructured tetrahedral meshes as well as local mesh refinements, the presented solver is able to represent complex geometries of subsurface structures very precisely and thus improve the solution accuracy and avoid misleading artefacts in images. Consequently, it can be successfully used in geological environments of arbitrary geometrical complexities. The parallel implementation of the method, which is based on the domain decomposition and a hybrid MPI–OpenMP scheme, has proved to be highly scalable – the achieved speed-up is close to the linear for more than a thousand processors. As a result, the code is able to deal with extremely large problems, which may have hundreds of millions of degrees of freedom, in a very efficient way. The importance of having this forward-problem solver lies in the fact that it is now possible to create a 3-D electromagnetic inversion that can deal with data obtained in extremely complex geological environments in a way that is realistic for practical use in industry. So far, such imaging tools have not been proposed due to a lack of efficient, parallel finite-element solutions as well as the limitations of efficient solvers based on finite differences.

In addition, the thesis discusses physical, mathematical and numerical aspects and challenges of 3-D electromagnetic modelling, which have been studied during my research in order to properly design the presented software for electromagnetic field simulations on 3-D areas of the Earth. Through this work, a physical problem formulation based on the secondary Coulomb-gauged electromagnetic potentials has been validated, proving that it can be successfully used with the standard nodal finite-element method to give highly accurate numerical solutions. Also, this work has shown that Krylov subspace iterative methods are the best solution for solving linear systems that arise after finite-element discretisation of the problem under consideration. More precisely, it has been discovered empirically that the best iterative method for this kind of problems is biconjugate gradient stabilised with an elaborate preconditioner. Since most commonly used preconditioners proved to be either unable to improve the convergence of the implemented solvers to the desired extent, or impractical in the parallel context, I have proposed a preconditioning technique for Krylov methods that is based on algebraic multigrid. Tests for various problems with different conductivity structures and characteristics have shown that the new preconditioner greatly improves the convergence of different Krylov subspace methods, even in the most difficult situations, which significantly reduces the total execution time of the program and improves the solution quality. Furthermore, the preconditioner is very practical for parallel implementation. Finally, through this work, it has been concluded that there are not any restrictions in employing classical parallel programming models, MPI and OpenMP, for parallelisation of the presented finite-element solver. Moreover, these programming models have proved to be enough to provide an excellent scalability for it, as shown by different large-scale tests.

Keywords: high-performance computing, parallel programming, 3-D electromagnetic modelling, finite element, preconditioning, algebraic multigrid

To my parents.

Acknowledgements

First of all, I would like to thank my thesis advisor, José María Cela, for accepting me in his team in *Barcelona Supercomputing Center* and for proposing an inspiring and exciting research path, along which he has supported me with patience and understanding, as well as with complete readiness to share his broad knowledge. I would also like to thank all my colleagues and office mates from *BSC–Repsol Research Center*, who are not only great co-workers, but also wonderful people, for all the help with different aspects of my work, as well as for creating a relaxed, pleasant and cheerful working environment in the office 302 in *Nexus I*. I am especially grateful to Josep de la Puente, who appeared just in time to significantly facilitate my research, for sharing his knowledge as well as his clear and bright insights and, maybe even more, for simply being always there for an inspiring discussion. Also, he is the one of few people who patiently and thoroughly red my articles as well as this thesis and I greatly appreciate his useful comments and ideas that helped me to improve all my writings. I am also thankful to Vladimir Puzyrev for helping me a lot in the past few years by working on the same project and for sharing with me all ups and downs of the working process (and a few musical concerts, as well), Félix Rubio Dalmau and Albert Farres for always being ready to save me from my occasional struggles with computers and for being very good friends during all these years in Barcelona, as well as Mauricio Hanzich, Jean Kormann, Oscar Peredo, Miguel Ferrer, Natalia Gutiérrez, Genís Aguilar and Juan Esteban Rodríguez for being such a nice team. Here, I would also like to thank *Repsol* for financially supporting the research whose result is the presented thesis.

Also, I am very thankful to all my colleagues from *Computer Applications in Science and Engineering* department of *Barcelona Supercomputing Center* that supported me during my research and shared with me their insight and expertise on different aspects of the topic. I am especially grateful to Guillaume Houzeaux for finding the time in his incredibly busy daily schedule to help me to cope with the *Alya* system and Xavie Sáez for his valuable insights into parallelisation methods and issues as well as for being a great friend that is always ready to explore

different artistic and cultural events. I would also like to acknowledge all other members of *Barcelona Supercomputing Center* that contributed in different ways to my academic life in Barcelona. Here, above all, I have to thank Mateo Valero for accepting me six years ago into this wonderful, stimulating and vibrant environment and for giving me a chance to take up this exciting and life-changing academic journey. However, none of this would have happened if there had not been for Professor Veljko Milutinović from University of Belgrade, who believed in me enough to give me his recommendation which opened me the door to this new world of incredible opportunities. Also, I would like to acknowledge former and current members of the Serbian community in *BSC* (which is not small) that helped me to overcome numerous practical difficulties that foreign students normally face as well as to feel less homesick, thanks to which I was able to primarily focus on my research. Especially, I would like to thank Nikola Marković, Vesna Nowack, Ana Jokanović, Maja Etinski, Srdjan Stipić and all Vladimirs for helping me at different points of my academic and non-academic life in Barcelona to overcome some difficult times and to find a way to carry on.

Outside *BSC*, there are people from other academic communities that I would like to mention. First, I would like to thank Xavier García from Barcelona Center for Subsurface Imaging (Barcelona-CSI) for sharing his knowledge of geophysical explorations with electromagnetic methods and for introducing me to the community of electromagnetic geophysicists at the unforgettable workshop on magnetotelluric 3-D inversion, held in Dublin in March 2011, at which I have made a lot of valuable contacts, learned a lot and found inspiration to continue with my research, In May 2012, I visited the leading scientist in the community, Gregory Newman, and his famous group at the Geophysics Department in the Earth Sciences Division of Lawrence Berkeley National Laboratory, Berkeley, CA, USA. I am extremely grateful to Dr. Newman for his willingness to be my host and to share his incredible knowledge of the topic. Also, I would like to express deep gratitude to Evan Um, a great scientist and a wonderful person, for agreeing to share his office with me during my visit as well as for helping me to improve my work by sharing his broad knowledge and experience with unique patience and kindness.

Barcelona is a vibrant and fascinating city with many opportunities for young people. Therefore, being a PhD student in it is both exciting and challenging. Luckily, I have managed to dedicate myself to my research without losing too much of experiencing life outside the office. I would like to thank Irina Blagojević, Jelena Lazarević,

Neda Kostandinović, Ognjen Obućina and Rade Stanojević for exploring this city with me and for sharing some wonderful and unforgettable moments, as well as for broadening my views by talking about other interesting topics from outside of the worlds of computer architecture and geophysics. My greatest support and source of strength and love in Barcelona during the past few years has been Vladimir Gajinov, who shared with me the best and the worst moments and who helped me to reorganise my priorities and to focus more efficiently on my research. For all of that, and much more, I am deeply grateful to him.

I would also like to thank my whole wonderful family and all my friends in Serbia and other places outside Barcelona that stayed close to me despite physical distances between us. Here, I would like to especially thank Sonja Veselinović, Tatjana Dosković, Ljiljana Popović, Andjelija Malenčić, Dušica Čolović, Vesna Babić, Ana Ljubojević, Maja Krneta, Tamara Tošić, Milica Paut, Bojana Vujin, Mina Ivović, Sandra Knežević, Slavko i Danica Ivović, Maja i Mara Graorinov for regularly being on Skype or writing mails, as well as for always finding the time to meet me whenever we found ourselves in the same city. Their friendship and support in every situation is my most valuable non-academic achievement that I am extremely proud of. Finally, I owe endless gratitude to my parents, my father Milenko and my mother Desanka, as well as my aunt Jadranka Ivović and grandmother Ljubinka Koldan, who loved me enough to let me go and who have been a priceless source of love, strength and happiness that has always been there for me unconditionally.

Contents

List of Figures

List of Tables

Preface

The presented PhD thesis is the result of the research that has been carried out at the *Repsol-BSC Research Center*. The objective of the thesis is the development of an efficient, robust and reliable numerical method for 3-D electromagnetic modelling in exploration geophysics.

Electromagnetic (EM) methods, such as magnetotellurics (MT) and controlled-source EM (CSEM) methods, have been increasingly used for oil and gas exploration thanks to their growing significance and technical advancement. The spreading employment of EM methods in exploration geophysics have led to the increasing need for reliable and fast techniques of interpretation of 3-D EM data sets acquired in extremely complex geological environments. However, due to the fact that industrial large-scale surveys need to collect immense amounts of data in order to obtain realistic subsurface images of huge Earth areas, the solution of the 3-D EM inverse problem is immensely challenging. Even with the very high level of modern computing technology, the proper numerical solution of this problem still remains a computationally extremely demanding task. Consequently, there are very few efficient solutions to this problem and only one practical, highly efficient, fully parallel 3-D CSEM inversion code, developed by Newman & Alumbaugh (1997). Because of this, the industry normally employs 2.5-D, 2-D, or even 1-D, programs to interpret responses arising from 3-D geologies, which naturally leads to incomplete or false interpretations. Moreover, most of the existing solutions, including the one of Newman & Alumbaugh (1997), are based on a numerical technique that cannot accurately take into account complex subsurface geometries, which can lead to misinterpretations in many situations. A lack of practical inversion schemes, due to enormously high computational requirements, as well as the limitations of the most commonly used numerical approach, make the main obstacles to wider and more frequent industrial applications of EM methods. This was the motivation for *Repsol* to initiate and financially support the development of very fast and reliable CSEM imaging tools that can deal with any situation faced in practical use.

The first and most important step to creating an inversion method is the development of a solver for the forward problem. Namely, one of the reasons for the huge computational demands of 3-D EM inversion is the expensive solution of the 3-D EM forward problem which

is, in addition, solved many times within inversion algorithms to simulate the EM field. The forward-problem solution is expensive because usually it is necessary to calculate hundreds of millions of field unknowns. Also, normally it is needed to solve thousands forward problems within an inversion algorithm just for one imaging experiment. It is clear that in order to create an efficient, reliable and practical interpretation method for 3-D EM data acquired in increasingly complex geological environments, it is necessary to have an accurate, very fast and robust 3-D modelling. As already said, this thesis focuses precisely on this extremely important and demanding step to building a 3-D EM interpretation method.

3-D EM modelling, i.e. EM field simulation on a 3-D area of the Earth, involves numerical solution of the diffusive Maxwell's equations in heterogeneous anisotropic electrically conductive media. In order to create this solution, the first important decision that has to be made is the choice of a numerical method for discretisation of the original continuous problem described by partial differential equations. The most popular approach is finite difference (FD) and most of the existing solvers are based on it. However, this method supports only structured rectangular grids which can be a big limitation in many situations – e.g. complex geological structures that affect measurements cannot be accurately modelled, which can lead to wrong interpretations. Also, local grid refinements are not supported and, consequently, it is not possible to have a finer mesh at some place without increasing the overall computational requirements. The alternative to FD is the finite-element (FE) approach. The main advantage of this method is that is supports completely unstructured meshes and thus is able to take into account arbitrary complex and irregular geometries more accurately than other techniques, which is important to avoid misleading artefacts in images. Also, it allows local mesh refinements, which means that it is possible to have small elements just in the places where a better resolution is required, without increasing already huge computational demands. The problem with this method is that its classical nodal form cannot be applied for the natural problem formulation in terms of the vector EM-field functions. Therefore, some modifications of the method are necessary. Because of this, most researchers focused on overcoming this obstacle in order to create an accurate and reliable FE EM forward-problem solver, and much less effort has been put in the efficiency improvement. Consequently, there are no truly efficient, fast, parallel FE solutions that would make 3-D FE inversion more realistic for practical use in industry. As previously commented, the only completely parallel and very fast and efficient solver is based on finite differences. In order to be able to accurately deal with arbitrary subsurface geometries, and not to be limited by structured rectangular grids, I have decided to create a numerical method for 3-D CSEM modelling based on finite elements. Due to the fact that the main purpose of this solver is to be a critical part of an inversion method, the main focus of my research has been to find the

most efficient elements for the final numerical scheme as well as to improve the efficiency of that scheme by its implementation on massively parallel computing platforms.

Having decided to use the FE approach, I have had to deal with the challenge that arises from the necessity to modify the standard nodal version of the method. Therefore, my next task has been to find a proper modification. Considering that I have decided not to modify the node-based elements themselves, I have had to modify the formulation of the physical problem and to use a formulation based on the so-called secondary Coulomb-gauged EM potentials. Having chosen the numerical method and the proper problem formulation, the next step has been to deal with the resultant system of linear equations that is a product of the discretisation of the original problem by finite elements. Solution of this system is the most time-consuming part of the code ($\sim 90\%$ of the overall execution time) and therefore demands a special attention. Obviously, it is critical to find the most appropriate and most efficient numerical solver. It is well known that for large-scale problems, such as the one under consideration, the best choice are iterative methods because of their low memory requirements and more efficient parallelisation. The problem with this group of linear solvers is that they are not generic and that there are no mathematical rules that can tell us which particular method is the best choice for the problem. Therefore, it is necessary to implement and test several different options in order to discover the most suitable one. The most commonly used iterative solvers are Krylov subspace methods since they are generally very efficient. Therefore, I have implemented three different Krylov methods that are normally used for the problems of the same type as the one under consideration. The tests have shown that the best choice for this problem is biconjugate gradient stabilised. However, Krylov methods are known to converge quite badly in real large-scale applications. Furthermore, in this particular problem, the convergence is normally very poor due to high conductivity contrasts in models and big maximal size ratios of unstructured meshes. Therefore, it has been necessary to add a preconditioner that improves the convergence of the solver, so that it can reach prescribed precisions. Since tests have shown that usually employed preconditioners do not help in most situations, I have had to find a more elaborate one in order to enable convergence of the solver to needed precisions. In addition, this preconditioner has had to be practical in the parallel context. During this research, I came across the idea that algebraic multigrid (AMG) can be a very good preconditioner. Hence, I have implemented one simplified version of AMG that has proved to be a very powerful preconditioning scheme able to improve the convergence of the solver even in most difficult cases. This particular implementation has never been used in this kind of applications. Also, it has proved to be effective within the parallel implementation of the code.

Very efficient way to deal with immense computational demands that appear in practical applications is to design the numerical scheme for running on massively parallel computers. Since this has not been done before for 3-D CSEM solvers based on finite elements, I have started my research with the classical parallel programming models – MPI and OpenMP. I have developed a hybrid MPI–OpenMP parallel scheme based on the domain decomposition. The results of large-scale tests have shown that this parallel implementation of the method is highly scalable. Based on these results, I have concluded that these classical models work very well for this kind of problems and that there is no need to search for other parallel solutions.

Organisation of the thesis

Chapter 1 of the thesis first discusses the importance of 3-D EM inversion in modern exploration geophysics, as well as the importance of solving 3-D EM forward problem as efficiently as possible. Then, starting from the fundamental EM theory, the physical aspect of 3-D EM modelling, referred to as the physical problem in induction geophysics, is introduced. It is important to emphasise that the thesis deals with 3-D EM problems that appear in real geophysical explorations. After this, I discuss general approach to numerical solution of this problem where state-of-the-art numerical techniques used in the latest EM forward-problem numerical solvers are presented.

In Chapter 2, a parallel nodal finite-element solver that I have developed is described. First, I present a potential-based problem formulation that is used and validated in this work. Then, I go step-by-step through the development process describing all the numerical components that comprise the solver. This chapter also presents a novel elaborate preconditioning technique based on algebraic multigrid (AMG) which has been implemented in order to improve convergence of Krylov subspace methods and thus increase accuracy and efficiency of the whole numerical scheme.

Chapter 3 explains the parallel implementation of the presented numerical method. The parallelisation strategy is based on the domain decomposition (mesh partitioning) technique using the Message Passing Interface (MPI) programming paradigm for communication among computational units. I have implemented both synchronous and asynchronous versions of MPI communication. In addition to this, OpenMP is used for parallelisation inside of each computational unit. In this way I have created a hybrid MPI–OpenMP parallel scheme.

Chapter 4 presents results of different tests that have been carried out in order to evaluate the presented numerical solver. First, I confirm its accuracy through a set of simulations on different synthetic Earth models. I remark that I could not have used real data in this thesis

since they are strictly confidential property of the company. Also, I have confirmed that the method can be used for modelling different controlled-source and magnetotelluric problems in anisotropic media. Due to the fact that it supports completely unstructured tetrahedral meshes as well as mesh refinement, it is possible to represent complex geological structures very precisely and thus improve the solution accuracy. Next, the AMG preconditioning scheme has been validated through variety of tests. These tests have proved that the AMG preconditioner improves convergence of Krylov subspace methods and increases both accuracy and efficiency of the whole numerical scheme to a great extent. Finally, scalability tests on massively parallel computers have shown that the code is highly scalable – the achieved speed-up is close to the linear for more than a thousand processors.

In Chapter 5, I conclude that the presented finite-element solver is numerically stable and gives highly accurate solutions. Also, its efficiency has been improved to a great extent by designing the algorithm to run on massively parallel computing platforms and by developing a new elaborate preconditioning scheme based on powerful algebraic multigrid. In this way, I have developed an accurate, robust, highly scalable and very fast code that can cope with extremely large problems, which may have hundreds of millions of degrees of freedom, in a very efficient way. This chapter also discusses the future work that concerns the development of an efficient and reliable 3-D EM inversion based on the described forward-problem solver.

Chapter 6 presents a list of publications on the topic.

Chapter 1

Electromagnetic Modelling in Geophysics

Exploration geophysics is the applied branch of geophysics that uses various physical measurements to obtain information about the subsurface of the Earth. By using surface methods, geophysicists measure physical properties at the surface of the Earth in order to detect or infer the presence and position of ore minerals, hydrocarbons, geothermal reservoirs, groundwater reservoirs and other buried geological structures. The ultimate goal of a geophysical analysis, in the context of geophysical exploration, is to build a constrained model of geology, lithology and fluid properties based upon which commercial decisions about reservoir exploration, development and management can be made. To achieve this, the Earth can be examined with a number of tools, such as seismic methods, controlled-source electromagnetic methods, magnetotellurics, well-logging, magnetic methods, gravity methods, etc. Each of these techniques obtains a corresponding data type that must be interpreted (inverted) within an integrated framework, so that the resultant Earth model is consistent with all the data used in its construction.

Among all the above-mentioned methods, seismic ones have a special place. It is widely accepted that seismic techniques are extremely powerful and generally applicable. They have become the hydrocarbon industries' standard method for obtaining high-resolution images of structure and stratigraphy which can guide exploration, appraisal and development projects. However, there are some situations in which seismic data fail to answer geophysical questions of interest. In those cases, complementary sources of data must be used to obtain the required information. For example, seismic methods are very effective at mapping geological reservoir formations. On the other hand, seismic properties have extremely poor sensitivity to changes in the type of fluids, such as brine, water, oil and gas. Because of this, in many situations it is difficult, or even impossible, to extract information about fluids trapped in the subsurface

from seismic data. The fact that the established seismic methods are not good at recognising different types of reservoir fluids contained in rock pores has encouraged the development of new geophysical techniques that can be combined with them in order to image fluids directly. A range of techniques, which have appeared recently and have shown a considerable potential, use electromagnetic (EM) waves to map variations in the subsurface electric conductivity, σ (S/m), or its reciprocal, $1/\sigma$ (Ωm), called electric resistivity, ρ, because conductivity/resistivity measurements show a high degree of sensitivity to the properties and distribution of fluids in a structure. For example, the resistivity of an oil-saturated region of a reservoir can be 1–2 orders of magnitude higher than the resistivity of the surrounding water-saturated sediments. Therefore, an increase in resistivity, in comparison with the resistivity values of the surrounding geological strata, may directly indicate potential hydrocarbon reservoirs. If we take into account this significant contrast between the resistivities of hydrocarbons and fluids like brine or water, as well as the fact that EM methods allow remote mapping of the subsurface electric conductivity or resistivity, it is clear that these methods are very useful and desirable for detecting hydrocarbon locations. In addition, the conductivity/resistivity information from EM surveys is complementary to seismic data and can improve constraints on the fluid properties when used in an integrated geophysical interpretation. This is just an example of why EM techniques have come to exploration geophysics to stay, and furthermore, of why they have been gaining increasing significance over the past decades.

In general, the use of EM exploration methods has been motivated by their ability to map electric conductivity, dielectric permittivity and magnetic permeability. The knowledge of these EM properties is of great importance since they can be used not only in oil and gas exploration, but also in hydrological modelling, chemical and nuclear waste site evaluations, reservoir characterisation, as well as mineral exploration. Nowadays, there is a great diversity of EM techniques, each of which has some primary field of application. However, many of them can be used in a considerably wide range of different fields. For example, EM mapping (Cheesman *et al.*, 1987), on land, produces a resistivity map which can detect boundaries between different types of rocks and directly identify local three-dimensional targets, such as base-metal mineral deposits, which are much more conductive than the host rocks in which they are found. This method is also used as a tool in the detection of sub-sea permafrost, as well as as a supplementary technique to seismic methods in offshore oil exploration. Furthermore, physical properties like porosity, bulk density, water content and compressional wave velocity may be estimated from a profile of the subsurface electric conductivity with depth.

The marine controlled-source electromagnetic (CSEM) method is nowadays a well-known geophysical exploration tool in the offshore environment and a commonplace in industry (e.g.

Constable & Srnka, 2007; Li & Key, 2007; Constable, 2010; Weitemeyer *et al.*, 2010). Marine CSEM, also referred to as seabed logging (Eidesmo *et al.*, 2002), explores the sub-seabed conductivity structure by emitting a low-frequency EM wave from a high-powered source (normally an electric dipole) which is connected to a vessel and towed close to the seabed. The transmitted wave (primary field) interacts with the electrically conductive Earth and induces eddy currents that become sources of a secondary EM field. The two fields, the primary and the secondary one, add up to a resultant field, which is measured by remote receivers placed on the seabed. Since the secondary field at low frequencies, for which displacement currents are negligible, depends primarily on the electric conductivity distribution of the ground, it is possible to detect thin resistive layers beneath the seabed by studying the received signal. Operating frequencies of transmitters in CSEM may range between 0.1 and 10 Hz, and the choice depends on the dimensions of a model. In most studies, typical frequencies vary from 0.25 to 1 Hz, which means that for source-receiver offsets of 10–12 km, the penetration depth of the method can extend to several kilometres below the seabed (Andréis & MacGregor, 2008). Moreover, CSEM is able to detect resistive layers that are very thin – only a few tens of meters thick. Fig. 1.1 shows the marine controlled-source electromagnetic method in combination with the marine magnetotelluric (MT) method. The difference between CSEM and MT is that MT techniques

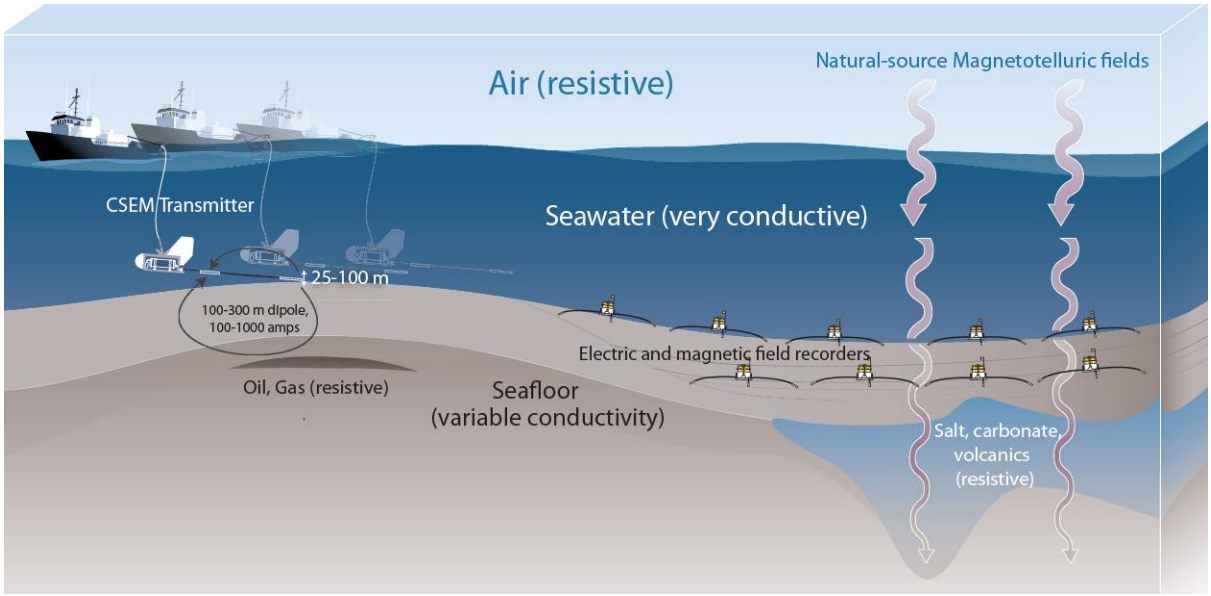


Figure 1.1: Marine EM.

use natural, airborne, transmitters instead of active, man-made, sources employed in CSEM. Namely, in MT, the source origin is in the ionosphere and the input wave is a plane wave. Also, CSEM is more sensitive to resistors while MT is sensitive to conductors. The marine CSEM

method has long been used to study the electric conductivity of the oceanic crust and upper mantle. However, more recently, an intense commercial interest has arisen to apply the method to detect offshore hydrocarbon reservoirs. This is due to the fact that CSEM works best in deep water, where drilling is especially expensive. Also, the method has proven effective for characterisation of gas hydrate-bearing shallow sediments. Moreover, during the last decade, CSEM has been considered as an important tool for reducing ambiguities in data interpretation and reducing the exploratory risk. The academic and industrial development of the method is discussed in the review paper by Edwards (2005) and the recent paper by Key (2012).

1.1 3-D Electromagnetic Inversion

The growing significance of EM methods, such as CSEM and MT, has led to huge improvements in both instrumentation and data acquisition techniques and thus to the increasing employment of these methods in practice. With new and improved acquisition systems, it has become possible to acquire large amounts of high-quality EM data in extremely complex geological environments. Concurrently with this advance in data acquisition technology, a significant progress has been made in data processing capabilities, as well. Thanks to all these advancements, EM surveys are now designed to acquire data along several parallel profiles rather than along only one or two like in traditional approaches. The use of multiple lines leads to much better delineation of 3-D geological structures.

On the other hand, in parallel with these developments of EM methods, computer technology has undergone huge improvements of its own, especially in terms of speed and memory capabilities. This has allowed the development of algorithms that more accurately take into account some of the multi-dimensionality of the EM interpretation problem. For example, two-dimensional EM inversion schemes that 10 years ago required a Cray computer for reasonable computation times, now can finish execution in a few minutes up to an hour on standard desktop workstations and PCs. Also, computationally efficient algorithms have been developed that either make subtle approximations to the 2-D problem (Smith & Booker, 1991; Siripunvaraporn & Egbert, 2000) or use efficient iterative gradient algorithms (Rodi & Mackie, 2001) to produce 2-D images of geological structures.

In 3-D environments, 2-D interpretation of data is a standard practice due to its short processing times as well as the fact that there are very few efficient 3-D EM modelling and inversion schemes. However, in many 3-D cases, the use of 2-D interpretation schemes may give images in which appear some artefacts that could lead to misinterpretation. Therefore, there is a real and increasing need for reliable and fast techniques of interpretation of 3-D EM data

sets acquired in complex geological environments. In other words, it is extremely important to develop efficient and robust 3-D inversion algorithms. However, EM interpretation process, also referred to as EM inversion, is immensely challenging. Even with the very high level of modern computing technology, the proper numerical solution of the 3-D inverse problem still remains a very difficult and computationally extremely demanding task for several reasons. And precisely those enormously high computational requirements make the main obstacle to wider and more frequent industrial applications of EM methods.

One of the reasons for the huge computational demands of 3-D EM inversion is the expensive solution of the 3-D EM modelling problem, which is, in addition, solved many times within inversion algorithms to simulate the EM field. This is a consequence of the fact that, nowadays, geophysical exploration with EM methods extend to extremely complex geological environments, which requires a modelling solver to be able to correctly simulate the responses arising from realistic 3-D geologies. However, the modelling grids designed to approximate large-scale complex geologies, which include structures with complicated shapes and big contrasts in conductivities, are usually enormous, and hence computationally too expensive to allow fast forward simulations (normally, for realistic, smooth and stable 3-D reconstructions, it is necessary to solve up to hundreds of millions of field unknowns in the forward problem). Moreover, in order to obtain realistic subsurface images, which have a sufficient level of resolution and detail, of huge Earth areas, industrial large-scale surveys need to collect vast amounts of data. Those huge data sets require, in addition to massive modelling grids, a large number of forward-problem solutions in an iterative inversion scheme. For example, present-day exploration with the CSEM technology in search for hydrocarbons in highly complex and subtle offshore geological environments need to employ up to hundreds of transmitter-receivers arrays, which operate at different frequencies and have a spatial coverage of more than 1000 km^2 (Commer *et al.*, 2008). Also, industrial CSEM data is usually characterised by a large dynamic range, which can reach more than ten orders of magnitude. This requires the ability to analyse it in a self-consistent manner that incorporates all structures not only on the reservoir scale of tens of metres, but also on the geological basin scale of tens of kilometres. Besides this, it is necessary to include salt domes, detailed bathymetry and other 3-D peripheral geology structures that can influence the measurements. Typically, industrial CSEM data sets are so enormous that their 3-D interpretation requires thousands of solutions to the forward problem for just one imaging experiment. Clearly, the ability to solve the forward problem as efficiently as possible is essential to the strategies for solving the 3-D EM inverse problem. Taking everything into account, the conclusion is that 3-D EM inversion needs a fast, accurate and reliable 3-D EM forward-problem solver in order to improve its own efficiency and thus to be realistic for practical use in industry.

Consequently, great strides have been made in geophysical 3-D EM modelling using different numerical methods. For example, approximate forward schemes (e.g. Zhdanov *et al.*, 2000; Habashy, 2001; Tseng *et al.*, 2003; Zhang, 2003) may deliver a rapid solution of the inverse problem, especially for models with low conductivity contrasts, but the general reliability and accuracy of these solutions are still open to question. Also, there are quite efficient forward solutions based on staggered 3-D finite differences (Yee, 1966; Druskin & Knizhnerman, 1994; Smith, 1996; Wang & Hohmann, 1993; Newman, 1995). All of them employ some kind of staggered finite-difference grid to calculate EM fields in both the time and/or frequency domain.

Nevertheless, even with these computationally efficient solutions, the complexity of models that can be simulated on traditional serial computers is limited by the flop rate and memory of their processors. Therefore, implementation of a 3-D inversion algorithm that uses these solutions is not practical on such machines. However, with the recent development in massively-parallel computing platforms, the limitations posed by serial computers have been disappearing. This is due to the fact that the rate at which simulations can be carried out is dramatically increased since thousands of processors can operate on the problem simultaneously. Thanks to this tremendous computational efficiency, it is possible to propose an efficient and practical solution of the 3-D inverse problem. Newman & Alumbaugh (1997) describe so far the only massively parallel, and thus truly efficient and practical, 3-D CSEM inversion, which uses parallel finite-difference forward-modelling code presented by Alumbaugh *et al.* (1996). This inversion scheme has been successfully applied to various real data (Alumbaugh & Newman, 1997; Commer *et al.*, 2008) which have been inverted in reasonable times. These works have also reported the benefits from using massively-parallel supercomputers for 3-D imaging experiments.

1.2 3-D Electromagnetic Modelling

3-D EM modelling, i.e. EM field simulation on a 3-D area of the Earth, is used today not only as an engine for 3-D EM inversion, but also for verification of hypothetical 3-D conductivity models constructed using various approaches and as an adequate tool for various feasibility studies. Some well-known EM methods used in exploration geophysics, such as CSEM and MT, highly depend on a good understanding of EM responses in arbitrary 3-D geological settings. In order to improve this understanding, it is essential to be able to model EM induction in random 3-D electrically conductive media. In other words, it is necessary to have a tool that can accurately and efficiently determine EM responses to CSEM or plane-wave excitations of a 3-D electrically conductive inhomogeneous anisotropic area of the Earth.

1.2.1 Fundamental Electromagnetism

In order to comprehend the bases and interpretative techniques of EM prospecting methods, it is necessary to have some knowledge of EM theory (Nabighian, 1987).

Maxwell's Equations in the Time Domain

Experiments have showed that all EM phenomena are governed by empirical Maxwell's equations, which are uncoupled first-order linear partial differential equations (PDEs).

An EM field may be defined as a domain of four vector functions:

$\mathbf{e}$ (V/m) - electric field intensity,

$\mathbf{b}$ (Wb/m^2 or Tesla) - magnetic induction,

$\mathbf{d}$ (C/m^2) - dielectric displacement,

$\mathbf{h}$ (A/m) - magnetic field intensity.

As already mentioned, all EM phenomena obey Maxwell's equations whose form in the time domain is:

$$\nabla \times \mathbf{e} + \frac{\partial \mathbf{b}}{\partial t} = 0, \tag{1.1}$$

$$\nabla \times \mathbf{h} - \frac{\partial \mathbf{d}}{\partial t} = \mathbf{j}, \tag{1.2}$$

$$\nabla \cdot \mathbf{b} = 0, \tag{1.3}$$

$$\nabla \cdot \mathbf{d} = \rho, \tag{1.4}$$

where $\mathbf{j}$ (A/m^2) is electric current density and ρ (C/m^3) is electric charge density. This is the conventional general form of Maxwell's equations.

Constitutive Relations

Maxwell's equations are uncoupled differential equations of five vector functions: $\mathbf{e}, \mathbf{b}, \mathbf{h}, \mathbf{d}$ and $\mathbf{j}$. However, these equations can be coupled by empirical constitutive relations which reduce the number of basic vector field functions from five to two. The form of these relations in the frequency domain is:

$$\mathbf{D} = \tilde{\varepsilon}(\omega, \mathbf{E}, \mathbf{r}, t, T, P, \dots) \cdot \mathbf{E}, \tag{1.5}$$

$$\mathbf{B} = \tilde{\mu}(\omega, \mathbf{H}, \mathbf{r}, t, T, P, \dots) \cdot \mathbf{H}, \tag{1.6}$$

$$\mathbf{J} = \tilde{\sigma}(\omega, \mathbf{E}, \mathbf{r}, t, T, P, \dots) \cdot \mathbf{E}, \tag{1.7}$$

where tensors $\tilde{\epsilon}, \tilde{\mu}$ and $\tilde{\sigma}$ describe, respectively, dielectric permittivity, magnetic permeability and electric conductivity as functions of angular frequency, ω, electric field strength, $\mathbf{E}$, or magnetic induction, $\mathbf{B}$, position, $\mathbf{r}$, time, t, temperature, T, and pressure, P. In general case, each of these three tensors is complex and, consequently, the phases of $\mathbf{D}$ and $\mathbf{E}$, of $\mathbf{H}$ and $\mathbf{B}$ and of $\mathbf{J}$ and $\mathbf{E}$ are different.

It is very important to carefully choose the form of the constitutive relations that is suitable to describe the Earth in the problem that we want to solve. For example, in problems that arise in CSEM, it is normally assumed that the Earth is heterogeneous, anisotropic and with electromagnetic parameters that are independent of temperature, time and pressure.

Maxwell's Equations in the Frequency Domain

Maxwell's equations in the frequency domain are obtained by applying one-dimensional Fourier transformation on equations (1.1) and (1.2) and by using constitutive relations (1.5), (1.7) and (1.6):

$$\nabla \times \mathbf{E} - i\omega\tilde{\mu}\mathbf{H} = 0, \tag{1.8}$$

$$\nabla \times \mathbf{H} - (\tilde{\sigma} - i\omega\tilde{\varepsilon})\mathbf{E} = \mathbf{J}_S, \tag{1.9}$$

where ω is the angular frequency of the field with assumed time-dependence of the form: $e^{-i\omega t}$, $\mathbf{J}_S$ is the vector of the current density of a source, $\tilde{\sigma}\mathbf{E}$ is the ohmic conduction term and $i\omega\tilde{\varepsilon}\mathbf{E}$ is the term that describes displacement currents.

Potential Representations and Gauge Transformations

Many boundary-value problems can be solved in terms of the vector electric and magnetic field intensity functions. However, a boundary-value problem is often difficult to solve in terms of the vector field functions and is easier to solve using vector and/or scalar potential functions from which the vector field functions may be derived. Several different sets of potential functions appear in the literature.

Taking into account the fact that the divergence of the curl equals zero, $\nabla \cdot (\nabla \times \mathbf{A}) = 0$, equation (1.3) can be interpreted in such a way that vector function $\mathbf{b}$, which describes the

magnetic field, is considered to be the curl of some other vector function $\mathbf{a}$ and therefore can be derived from it:

$$\mathbf{b} = \nabla \times \mathbf{a}. \tag{1.10}$$

This new vector function $\mathbf{a}$ is called vector potential.

By inserting equation (1.10) into equation (1.1), we obtain:

$$\nabla \times (\mathbf{e} + \frac{\partial \mathbf{a}}{\partial t}) = 0. \tag{1.11}$$

Since the curl of the gradient is zero, $\nabla \times \nabla\phi = 0$, vector $\mathbf{e} + \dfrac{\partial \mathbf{a}}{\partial t}$ can be represented as some gradient:

$$\mathbf{e} + \frac{\partial \mathbf{a}}{\partial t} = -\nabla\phi, \tag{1.12}$$

where scalar function ϕ is called scalar potential.

From equation (1.12) follows:

$$\mathbf{e} = -\left(\nabla\phi + \frac{\partial \mathbf{a}}{\partial t}\right). \tag{1.13}$$

Consequently, both $\mathbf{e}$ and $\mathbf{b}$ can be represented using some potentials $\mathbf{a}$ and ϕ.

For any choice of $\mathbf{a}$ and ϕ, Maxwell's equations (1.1) and (1.3) are fulfilled. The identity

$$\mathbf{e} = -\left(\nabla\phi + \frac{\partial \mathbf{a}}{\partial t}\right) = -\left[\nabla(\phi - \frac{\partial \Psi}{\partial t}) + \frac{\partial(\mathbf{a} + \nabla\Psi)}{\partial t}\right] \tag{1.14}$$

shows that substitutions

$$\phi \to \phi' = \phi - \frac{\partial \Psi}{\partial t} \tag{1.15}$$

and

$$\mathbf{a} \to \mathbf{a}' = \mathbf{a} + \nabla\Psi \tag{1.16}$$

generate equivalent potentials ϕ' and $\mathbf{a}'$ for representation of $\mathbf{e}$ and $\mathbf{b}$, where Ψ is an arbitrary function. Also, ρ and $\mathbf{j}$, given by Maxwell's equations (1.2) and (1.4), remain unchanged by the above transformation. In other words, potentials ϕ and $\mathbf{a}$ are not unique. However, the values of these potentials are not important. The important thing is that when differentiated according to Maxwell's equations, they result in the right fields $\mathbf{e}$ and $\mathbf{b}$. Choosing a value for ϕ and $\mathbf{a}$ is called choosing a gauge, and a switch from one gauge to another, such as going from ϕ and $\mathbf{a}$ to ϕ' and $\mathbf{a}'$, is called a gauge transformation with generating function Ψ. From what has been stated above, it is clear that the above gauge transformation leaves $\mathbf{e}$, $\mathbf{b}$, $\mathbf{j}$ and ρ invariant for an arbitrary function Ψ.

For a given set of densities ρ and $\mathbf{j}$, it is necessary to show the existence of potentials ϕ and $\mathbf{a}$ which fulfil Maxwell's equations (1.2) and (1.4). Inserting representations (1.10) and (1.12) into (1.2) yields:

$$\nabla \times \nabla \times \mathbf{a} + \frac{1}{c^2}\left(\frac{\partial^2 \mathbf{a}}{\partial t^2} + \nabla \frac{\partial \phi}{\partial t}\right) = \mu \mathbf{j} \tag{1.17}$$

and due to $\nabla \times \nabla \times = \nabla\nabla \cdot - \Delta$, follows:

$$\frac{1}{c^2}\frac{\partial^2 \mathbf{a}}{\partial t^2} - \Delta \mathbf{a} + \nabla\left(\nabla \cdot \mathbf{a} + \frac{1}{c^2}\frac{\partial \phi}{\partial t}\right) = \mu \mathbf{j}. \tag{1.18}$$

Analogously, by insertion of (1.12) into (1.4), we obtain:

$$-\left(\Delta \phi + \nabla \cdot \frac{\partial \mathbf{a}}{\partial t}\right) = \frac{\rho}{\varepsilon} \tag{1.19}$$

or

$$\frac{1}{c^2}\frac{\partial^2 \phi}{\partial t^2} - \Delta \phi - \frac{\partial\left(\nabla \cdot \mathbf{a} + \frac{1}{c^2}\frac{\partial \phi}{\partial t}\right)}{\partial t} = \frac{\rho}{\varepsilon} \tag{1.20}$$

Finally, it can be checked that the application of the gauge transformation transforms (1.18) and (1.20) into:

$$\frac{1}{c^2}\frac{\partial^2 \mathbf{a}'}{\partial t^2} - \Delta \mathbf{a}' + \nabla\left(\nabla \cdot \mathbf{a}' + \frac{1}{c^2}\frac{\partial \phi'}{\partial t}\right) = \mu \mathbf{j} \tag{1.21}$$

and

$$\frac{1}{c^2}\frac{\partial^2 \phi'}{\partial t^2} - \Delta \phi' - \frac{\partial\left(\nabla \cdot \mathbf{a}' + \frac{1}{c^2}\frac{\partial \phi'}{\partial t}\right)}{\partial t} = \frac{\rho}{\varepsilon}, \tag{1.22}$$

i.e. both equations are gauge invariant.

At this point, the existence of solutions ϕ and $\mathbf{a}$ of coupled differential equations (1.18) and (1.20), for a given set of ρ and $\mathbf{j}$, cannot be guaranteed. However, using the freedom of gauge transformations it can be showed that the equations can be transformed in a decoupled system that is solvable. This implies that equations (1.18) and (1.20) are solvable too.

Coulomb Gauge

As already mentioned, a problem in resolving equations (1.18) and (1.20) is their coupling. One way of decoupling them is by introducing an additional condition:

$$\nabla \cdot \mathbf{a} = 0, \tag{1.23}$$

i.e. term $\nabla \cdot \mathbf{a}$ in equation (1.20) has to be removed by applying an appropriate gauge transformation.

Since the gauge transformation performs the following:

$$\nabla \cdot \mathbf{a} \to \nabla \cdot \mathbf{a} + \Delta \Psi, \tag{1.24}$$

it is merely necessary to choose a generating function Ψ as a solution of:

$$\Delta\Psi = -\nabla \cdot \mathbf{a}. \tag{1.25}$$

This gauge transformation transforms equation (1.20) into:

$$\frac{1}{c^2}\frac{\partial^2\phi'}{\partial t^2} - \Delta\phi' - \frac{1}{c^2}\frac{\partial^2\phi'}{\partial t^2} = \frac{\rho}{\varepsilon} \tag{1.26}$$

or

$$\Delta\phi' = \frac{\rho}{\varepsilon}, \tag{1.27}$$

which is the Poisson equation with well-known integral solution.

Therefore, ϕ' is now a known function for the other equation which can be rewritten as:

$$\frac{1}{c^2}\frac{\partial^2\mathbf{a}'}{\partial t^2} - \Delta\mathbf{a}' = \mu\mathbf{j}^*, \tag{1.28}$$

where $\mathbf{j}^* = \mathbf{j} - \varepsilon\nabla\frac{\partial\phi}{\partial t}$ is a known function.

Again, solution of the above equation in integral form is known.

Boundary Conditions

Boundary conditions on the vector field functions can be obtained from the straight-forward application of the integral form of Maxwell's equations:

Normal J: the normal component, J_n, of $\mathbf{J}$ is continuous across the interface separating medium 1 from medium 2, i.e:

$$J_{n_1} = J_{n_2}. \tag{1.29}$$

Strictly speaking, this applies only to direct currents. However, it can be accepted completely if displacement currents may be neglected, like, for example, in the case of frequencies up to 10^5 Hz in the Earth materials.

Normal B: the normal component, B_n, of $\mathbf{B}$ is continuous across the interface separating medium 1 from medium 2, i.e:

$$B_{n_1} = B_{n_2}. \tag{1.30}$$

Normal D: the normal component, D_n, of $\mathbf{D}$ is discontinuous across the interface separating medium 1 from medium 2 due to accumulation of surface charges whose density is ρ_s, i.e:

$$D_{n_1} - D_{n_2} = \rho_s. \tag{1.31}$$

Tangential E: the tangential component, E_t, of $\mathbf{E}$ is continuous across the interface separating medium 1 from medium 2, i.e:

$$E_{t_1} = E_{t_2}. \tag{1.32}$$

Tangential H: the tangential component, H_t, of $\mathbf{H}$ is continuous across the interface separating medium 1 from medium 2 if there are no surface currents, i.e:

$$H_{t_1} = H_{t_2},\ J_s = 0. \tag{1.33}$$

1.2.2 Physical Problem in Exploration Geophysics

EM problems that arise in geophysical explorations when using methods like CSEM and MT generally deal with a resultant EM field which appears as a response of the electrically conductive Earth to an impressed (primary) field generated by a source. As already explained, the primary field gives rise to a secondary distribution of charges and currents and, hence, to a secondary field, and the resultant field is the sum of the primary and the secondary field. Each of the fields must satisfy Maxwell's equations, or equations derived from them, as well as appropriate conditions applied at boundaries between homogeneous regions involved in the problem, e.g. at the air-earth interface.

3-D CSEM modelling involves the numerical solution of Maxwell's equations in heterogeneous anisotropic electrically conductive media in order to obtain the components of the vector EM field functions within the domain of interest. Considering that one generally deals with stationary regimes, Maxwell's equations are most commonly solved in the frequency domain. Also, it is not needed to solve the general form of the equations due to the fact that CSEM methods normally use very low frequencies ($\sim$1 Hz). Namely, at low frequencies, displacement currents, $i\tilde{\varepsilon}\omega\mathbf{E}$, can be neglected, since the electric conductivity of geological structures is much larger than their dielectric permittivity, $\tilde{\sigma} \gg \tilde{\varepsilon}$, and therefore the general Maxwell's equations simplify and reduce to the diffusive Maxwell's equations:

$$\nabla \times \mathbf{E} = i\mu_0\omega\mathbf{H}, \tag{1.34}$$

$$\nabla \times \mathbf{H} = \mathbf{J}_S + \tilde{\sigma}\mathbf{E}, \tag{1.35}$$

where μ_0 is the magnetic permeability of the free space. In order to simplify the analysis, in most geophysical EM problems, it is assumed that all media posses electric and magnetic properties which are independent of time, temperature and pressure, as well as that magnetic permeability is a scalar and equal to the permeability of the free space, i.e. $\mu = \mu_0$. The assumption that

magnetic permeability is a constant can be made only if there are no metal objects in the ground, which is usually the case in hydrocarbon explorations. In isotropic media, also electric conductivity is taken as a scalar that is a function of all three spatial coordinates, $\sigma(\mathbf{r})$. On the other hand, in anisotropic cases, electric conductivity is described with a 3×3 tensor, $\tilde{\sigma}$, whose components also vary in all three dimensions. Here, the ohmic conduction term, $\tilde{\sigma}\mathbf{E}$, describes induced eddy currents inside the Earth.

1.2.3 Numerical Solution

In order to obtain a numerical solution to partial differential equations, it is necessary to discretise the equations, which are, by nature, continuous, using some discretisation technique. There are several different approaches to acquiring a numerical solution to the PDEs (1.34) and (1.35). The most commonly used ones are the finite-difference (FD) and finite-element (FE) methods.

Finite-Difference Method

The finite-difference (FD) method is the most commonly employed approach: (e.g. Yee, 1966; Mackie *et al.*, 1994; Alumbaugh *et al.*, 1996; Xiong *et al.*, 2000; Fomenko & Mogi, 2002; Newman & Alumbaugh, 2002; Davydycheva *et al.*, 2003; Abubakar *et al.*, 2008; Kong *et al.*, 2008; Davydycheva & Rykhlinski, 2011; Streich *et al.*, 2011). In this approach, electric conductivity, the vector EM field functions and Maxwell's differential equations are approximated by their finite-difference counterparts within a rectangular 3-D mesh of size $M = N_x \times N_y \times N_z$. This transforms the original problem described by PDEs to a resultant system of linear equations, $\mathbf{A}_{FD} \cdot \mathbf{x} = \mathbf{b}$, where system matrix $\mathbf{A}_{FD}$ is a large, sparse, complex and symmetric $3M \times 3M$ matrix, vector $\mathbf{x}$ is a $3M$ vector that contains the values of the EM field in the nodes of the mesh and $3M$-vector $\mathbf{b}$ represents the sources and boundary conditions. Weiss & Newman (2002) have extended this approach to fully anisotropic media. In the time domain, FD schemes have been developed by, for example, Wang & Tripp (1996), Haber *et al.* (2002), Commer & Newman (2004). The main attraction of the FD approach for EM software developers is rather simple implementation and maintenance of a code based on it, especially when compared with other approaches. However, this method supports only structured rectangular grids, which may be a substantial limitation in many situations. Namely, structures with complex geometry cannot be accurately modelled with rectangular elements and some boundary conditions cannot be properly accommodated by their formulation. It is known that this may have a serious impact on the quality of solutions, like, for example, in the case of seabed variations that also can greatly affect EM responses (Schwalenberg & Edwards, 2004). Furthermore, local mesh refinements are

not possible in structured grids and therefore any mesh size adaptation has a large effect on the overall computational requirements.

Finite-Element Method

The finite-element (FE) method has long been used in applied mathematics and solid mechanics. In geophysics, however, it has been employed for only a few decades, since the FE modelling of 3-D EM induction in geophysical prospecting applications was until recently beyond the capabilities of available computers. Nowadays, this task is easily performed on desktop workstations thanks to the rapid development and extremely high level of the modern computing technology. Consequently, many FE-based implementations of EM modelling have appeared (e.g. Zyserman & Santos, 2000; Badea *et al.*, 2001; MacGregor *et al.*, 2001; Key & Weiss, 2006; Li & Key, 2007; Franke *et al.*, 2007; Li & Dai, 2011). However, this approach is still not as widely used as finite difference and a major obstacle for its broader adoption is that the standard nodal FE method does not correctly take into account all the physical aspects of the vector EM field functions. In FE, the components of the vector EM field functions are approximated by the sum of some basic functions, which are usually polynomial functions. This decomposition produces a resultant system of linear equations, $\mathbf{A}_{FE} \cdot \mathbf{x} = \mathbf{b}$, where system matrix $\mathbf{A}_{FE}$ is large, sparse, complex and, in general, non-symmetric and vector $\mathbf{x}$ contains the coefficients of the polynomials which are sought using the Galerkin method. The main attraction of the FE approach for geophysicists is that it is able to take into account arbitrary geometries more accurately than other techniques. Namely, the FE method has the advantage of supporting completely unstructured meshes, which are more flexible in terms of geometry and therefore more precise in modelling irregular and complicated shapes that often appear in the real heterogeneous subsurface geology (shapes of ore-bodies, topography, cylindrical wells, seabed bathymetry, fluid invasion zones, etc.). This is important since imprecise modelling of complex shapes may result in misleading artefacts in images. In addition, FE supports local mesh refinements, which allow a higher solution accuracy without increasing the overall computational requirements. Namely, with local mesh refinements, it is possible to have smaller grid elements, i.e. a finer grid, only at places where a better resolution is required (around transmitters, receivers, target locations as well as large conductivity contrasts). Otherwise, it would be necessary to refine the entire mesh, which greatly increases the computational cost. As already said above, the main disadvantage of this approach is that the standard nodal FE method (Burnett, 1987) has to be modified in order to be used for EM problems. Namely, node-based finite elements cannot be used for EM problems formulated in terms of the vector electric and/or magnetic field functions, $\mathbf{E}$ or $\mathbf{H}$, which is a natural and physically meaningful problem formulation. This is due to the fact

that the nodal FE method does not allow the discontinuity of the normal component of the electric field at material interfaces. Also, node-based finite elements are not divergence free, because of which some spurious purely divergent field components can appear in the solution. One solution to these problems is to use a different problem formulation that is based on EM potentials – coupled vector-scalar potential functions, $\mathbf{A}$ and Φ. Both the vector magnetic potential function, $\mathbf{A}$, and scalar electric potential function, Φ, are continuous across the interfaces between different materials, which solves the problem of discontinuity. In order to prevent the appearance of spurious purely divergent modes, it is necessary to apply an additional condition, the Coulomb gauge condition, $\nabla \cdot \mathbf{A} = 0$, that enforces zero divergence of the vector potential function at element level. This potential-based formulation solves both problems and therefore allows the use of nodal finite elements. The other possible solution are specialised vector (edge) elements (Nédélec, 1986; Jin, 2002), which are used in some recent approaches to CSEM modelling (Mukherjee & Everett, 2011; Schwarzbach *et al.*, 2011; da Silva *et al.*, 2012; Um *et al.*, 2012). In case of edge elements, the unknowns are the tangential components of the electric field along the edges of elements. Consequently, these elements permit the normal component of the electric field to be discontinuous across material interfaces. Also, vector elements are divergence free by construction and hence cannot support spurious modes. Since edge elements solve both problems, the direct vector field formulation can be used with these elements. More about advantages and disadvantages of these two possible modifications of the FE method can be found in the paper by Badea *et al.* (2001).

Linear Algebraic Solver

Regardless of which approach is employed, the initial EM forward problem is always reduced to the solution of a system of linear equations:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1.36}$$

where $\mathbf{A}$ is the system matrix, $\mathbf{x}$ is the solution vector and $\mathbf{b}$ is the right-hand-side (RHS) vector. Namely, a discretisation of a differential equation, which is by nature continuous, produces a system of linear algebraic equations with a finite number of unknowns. This resultant system approximates the partial differential equation, hence its solution is an approximate, i.e. numerical, solution to the original continuous problem. The system matrix, $\mathbf{A}$, is normally very sparse (its elements are primarily zeros) since it is a result of the discretisation of a differential operator. Also, in real applications, this matrix is extremely large. Consequently, solving the large-scale linear system is the most important and most expensive part of the overall numerical

method. Normally, it takes up to 90% of the whole execution time. Naturally, special attention has to be put to this part of the code.

To solve the linear system, one can use some algebraic solver chosen taking into account relevant characteristics of the system matrix. The properties of matrix $\mathbf{A}$ are determined by the method applied to solve the forward problem (FD or FE). If the partial differential equation is three-dimensional, or two-dimensional involving many degrees of freedom per point, the derived linear system is, as already said, very large and sparse. Since the memory and computational requirements for solving such a system may seriously challenge even the most efficient direct solvers available today, the key lies in using iterative methods (Saad, 2003). The main advantage of iterative methods is their low storage requirement, which resolves the memory issue of direct methods. In addition, there is another very important benefit thanks to which iterative methods can cope with huge computational demands more readily then direct techniques. Namely, iterative methods are much easier to implement efficiently on high-performance parallel computers than direct solvers. Currently, the most common group of iterative techniques used in applications are Krylov subspace methods (Saad, 2003).

There are two important aspects regarding the discretisation of the initial continuous problem (Avdeev, 2005). The first one is how accurately the linear system represents Maxwell's equations. The second one is how well the system matrix, $\mathbf{A}$, is preconditioned. The latter issue arises from the fact that condition numbers, $\kappa(\mathbf{A})$, of unpreconditioned system matrices may be very large due to, for example, high conductivity contrasts in models. This is especially true for matrices that appear in FE solvers because high element size ratios in unstructured meshes considerably deteriorate their condition numbers. For such poorly conditioned systems, Krylov methods converge extremely slowly, if they converge at all. In order to overcome this problem, a variety of preconditioners have been designed and applied. The most popular preconditioning schemes employed within FD and FE methods are Jacobi, symmetric successive over-relaxation (SSOR) and incomplete LU factorisation (Saad, 2003). These preconditioners work quite well for medium and high frequencies, providing convergence of Krylov iterations. However, at low frequencies, more precisely, when the induction number is low, $\sqrt{\omega\mu_0\tilde{\sigma}}h \ll 1$, Maxwell's equations degenerate, which leads to some difficulties in convergence (h is the characteristic grid size). Therefore, some more elaborate preconditioners, which have proved to be much more efficient than traditional ones, have been presented. For example, the low induction number (LIN) preconditioner has been introduced and tested in Newman & Alumbaugh (2002) and Weiss & Newman (2003). Also, there are multigrid preconditioners described and employed in Aruliah & Ascher (2002), Haber (2004) and Mulder (2006).

Multigrid

Multigrid (Briggs *et al.*, 2000) is a numerical approach to solving large, often sparse, systems of equations using several grids at the same time. The essence of multigrid comes from two observations. The first observation is that it is quite easy to determine the local behaviour of the solution, characterised by oscillatory (high-frequency) components, while it is much more difficult to find the global components of the solution, which are smooth (low-frequency) components. The second observation is that the slowly varying smooth components can be accurately represented with fewer points on a coarser grid, on which they become more oscillatory and hence can be easily determined.

For a long time, multigrid (multilevel) methods (Trottenberg *et al.*, 2001) have been developed concurrently, but quite independently of general-purpose Krylov subspace methods and preconditioning techniques. However, recently, standard multigrid solvers have been very often combined with some acceleration methods, such as Krylov subspace techniques (CG, BI-CGSTAB, GMRES, etc.), in order to improve their efficiency and robustness. Several recent applications of multigrid in different EM problems are discussed in Everett (2012). The simplest approach is to employ complete multigrid cycles as preconditioners.

Algebraic multigrid (AMG) methods (Stuben, 2001), originally designed for creating standalone solvers, can be very good preconditioners, as well. This is due to the fact that AMG techniques, unlike other one-level preconditioners, work efficiently on all error components – from low-frequency to high-frequency ones. Taking this into account, instead of building a standalone AMG solver, which requires the very expensive set-up phase, it is generally more efficient to use AMG as preconditioning by employing, for example, an aggressive coarsening strategy. Also, AMG methods do not need any geometric information and thus can be used as black-box preconditioners with different iterative schemes, which gives them a big advantage over geometric multigrid techniques.

Chapter 2

Numerical Method for 3-D Electromagnetic Modelling

In order to obtain a numerical solution to the 3-D electromagnetic (EM) forward problem, it is necessary to discretise the governing equations, which are continuous partial differential equations (PDEs), using some discretisation technique, as already explained in Chapter 1, Subsection 1.2.3. The choice of the discretisation method is the first important decision that has to be made in order to develop a numerical scheme that efficiently produces an accurate solution to the problem. Two most commonly used discretisation techniques, finite difference and finite element, are briefly presented in Subsection 1.2.3. The finite-difference (FD) method, despite the disadvantage of not being able to precisely take into account complex geometries of subsurface structures, which in some cases may significantly damage the quality of a solution, is still the most widely employed discretisation scheme. Naturally, there are many successful FD solutions for the 3-D EM forward problem, including the only practical, highly efficient code, which is designed for running on massively parallel computing platforms and developed by Alumbaugh *et al.* (1996). However, all of these solvers are limited by the fact that they support only structured rectangular grids, which make them rather inadequate for situations in which irregular and complicated geology, such as seabed bathymetry, has a significant influence on measurements, because, in those cases, an imprecise representation of geological structures produces artefacts in images that can lead to false interpretations. On the other hand, the finite-element (FE) method supports completely unstructured tetrahedral meshes as well as mesh refinements, due to which it is able to represent complex geological structures very precisely and thus improve the solution accuracy. However, despite these very important advantages, the FE method is still not as broadly used as finite difference. A major reason for this is that the standard nodal form of the FE method cannot be used to discretise the governing equations in

terms of the vector EM field functions because it does not correctly represent all the physical aspects of these vector functions. Consequently, most of the researchers who have employed the FE approach for 3-D EM numerical modelling have been primarily focused on overcoming this problem, as well as on solving other physical and numerical challenges, in order to obtain a proper and accurate numerical solution. As a result of this effort, several successful FE solvers for the 3-D EM forward problem have appeared. However, much less effort has been put into the improvement of the efficiency of those FE codes despite the fact that the efficiency of a forward-problem solver is critical for its use inside of an inversion algorithm. As a consequence, a highly efficient, fully parallel FE implementation of 3-D EM numerical modelling still has not been proposed.

Taking all of the above into account, I have decided to employ the FE method (Burnett, 1987) as the discretisation technique in order to develop a highly accurate and robust numerical scheme for 3-D controlled-source electromagnetic (CSEM) modelling that is able to correctly take into account geological structures of arbitrary geometrical complexity. Also, considering that the main purpose of this forward-problem solver is to be a part of an inversion algorithm, the efficiency of the numerical scheme and of its implementation is an extremely important aspect, which has become the main focus of my research due to a lack of other truly efficient FE codes. In order to develop a very fast FE CSEM forward-problem solution, it is necessary to chose the best standard numerical components that ensure not only the accuracy, but also the efficiency of the solver. Furthermore, more elaborate and more powerful numerical methods have to be proposed and introduced in order to improve the efficiency of the overall scheme. Finally, having a very efficient numerical method, it has to be translated into a code that is as fast as possible, primarily by designing it to run on massively parallel computers.

To facilitate the task a bit, I have chosen to use the standard node-based FE method, so that I can reuse a database of different nodal finite elements that already existed within the *Alya* system. *Alya* system (Houzeaux *et al.*, 2009) is a computational mechanics code, which supports unstructured meshes made of different types of elements, developed in Barcelona Supercomputing Center. It has a modular architecture that allows a new numerical solver to be easily introduced in order to use some existing features and facilities of the system, such as nodal finite elements. Also, *Alya* is designed since its inception for large scale supercomputing, which allows parallelisation of any new solver. As already said, *Alya* architecture is modular. It consists of a kernel, modules and services, as shown in Fig. 2.1. The kernel is the essence of the system that starts and controls execution of a program. It contains functionalities used by all the modules and services. A module solves a set of PDEs describing a physical problem. A service is an utility that can be used by all the modules and the kernel. In order to build a

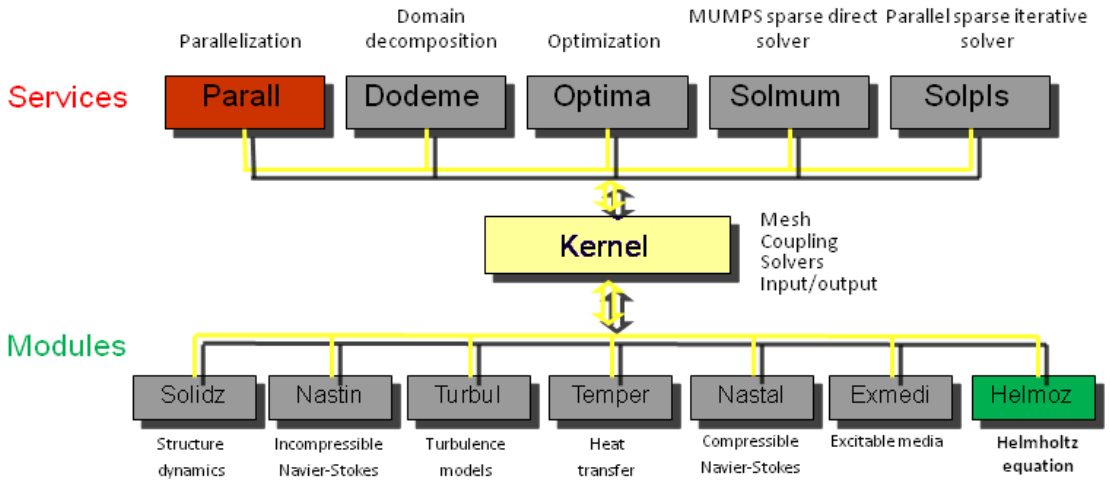


Figure 2.1: Architecture of the *Alya* system.

3-D CSEM solver, I had to create a new module for solving EM induction problems described by the diffusive Maxwell's equations (*Helmoz* module in Fig. 2.1). Also, I had to make some changes in the kernel, as well, so that *Alya* can support complex numbers. In addition, I have implemented new solvers and a new preconditioning scheme inside the kernel, so that they can be used by other modules, as well. Furthermore, I introduced new functionalities in the parallel service (*Parall* in Fig. 2.1) in order to build a more efficient parallel scheme, which is described in details in Chapter 3.

2.1 Physical Problem Formulation

As explained in Chapter 1, Subsection 1.2.3, the standard nodal FE method cannot be used if the EM problem, described by the diffusive Maxwell's equations, (1.34) and (1.35), is formulated in terms of the vector EM field functions, **E** and **H**. In order to be able to employ the node-based version of the method, it is necessary to use a different problem formulation that is based on some vector-scalar EM potential functions. In this work, the physical problem has been formulated in terms of the secondary Coulomb-gauged EM potentials (Badea *et al.*, 2001). This formulation solves not only the already mentioned problems of the electric-field discontinuity and spurious divergent modes, but also the problem of having singularities introduced by sources. In addition, it is numerically very stable.

Coulomb-Gauged Electromagnetic Potentials

The diffusive Maxwell's equations, (1.34) and (1.35), can be transformed if the EM field, (**E**, **H**), is expressed in terms of a magnetic vector potential, **A**, and an electric scalar potential, Φ, which are defined by:

$$\mathbf{B} = \nabla \times \mathbf{A}, \tag{2.1}$$

$$\mathbf{E} = i\omega\mathbf{A} - \nabla\Phi. \tag{2.2}$$

More information about these potentials is given in Chapter 1, Subsection 1.2.1.

Using expressions (2.1) and (2.2) to substitute the vector EM field functions by the EM potentials, equation (1.35) transforms into a curl-curl equation. The finite-element discretisation of this equation leads to asymmetric matrices which may cause numerical instabilities. In order to avoid this problem, I have followed the approach of Biro & Preis (1989), which transforms the curl-curl equation into the vector quasi-Helmholtz equation whose discretised form is numerically very stable:

$$\nabla^2\mathbf{A} + i\omega\mu_0\tilde{\sigma}(\mathbf{A} + \nabla\Psi) = -\mu_0\mathbf{J}_S, \tag{2.3}$$

where Ψ is the reduced scalar potential given by $\Phi = -i\omega\Psi$. However, in order to keep all physical conditions satisfied (a divergence-free current density), it is necessary to solve the auxiliary equation:

$$\nabla \cdot [i\omega\mu_0\tilde{\sigma}(\mathbf{A} + \nabla\Psi)] = -\nabla \cdot \mu_0\mathbf{J}_S \tag{2.4}$$

simultaneously with (2.3). Let me remark that equation (2.4) is more general than the one proposed by Badea *et al.* (2001) and it is valid for both inductively and directly coupled sources. This means that this formulation takes into account electric dipoles, which are the most common type of sources in geophysical applications. Equations (2.3) and (2.4) are valid inside the whole solution domain and constitute the incompletely gauged coupled vector-scalar potential formulation of Maxwell's equations. Nevertheless, these two equations alone are not sufficient to guarantee the uniqueness of the vector potential, **A**. Therefore, an additional condition, which is the Coulomb-gauge condition:

$$\nabla \cdot \mathbf{A} = 0, \tag{2.5}$$

must be applied in the whole solution domain. In order to apply this condition, it is enough to enforce the zero Dirichlet condition on $\nabla \cdot \mathbf{A}$ along the boundary of the solution domain.

Secondary Potentials

In electromagnetism, sources introduce singularities in their close proximity. Normally, this problem is solved by refining a mesh to a great extent around sources. However, this solution is quite costly in terms of computational requirements. Another approach that can be used to avoid singularities introduced by sources, and which is employed in this work, is the secondary potential formulation of the problem. A similar technique is usually used to account for a base state that does not belong to the finite-element space, and therefore cannot be represented exactly by the method. Instead of having it explicitly in the problem formulation through its current density, $\mathbf{J}_S$, a source of arbitrary shape, complexity and orientation can be introduced by dening a set of known primary EM potentials, ($\mathbf{A}_p$, Ψ_p). The primary potentials represent responses of either the homogeneous or horizontally layered Earth to the primary EM field transmitted by the source. They are normally determined by analytical expressions for EM induction in a homogeneous formation of a constant electric conductivity, $\sigma_p = const.$, or in horizontally layered models. The secondary EM potentials, ($\mathbf{A}_s$, Ψ_s), are defined by:

$$\mathbf{A} = \mathbf{A}_p + \mathbf{A}_s, \tag{2.6}$$

$$\Psi = \Psi_p + \Psi_s. \tag{2.7}$$

Primary Potentials

Nowadays, the most commonly used CSEM sources are horizontal electric dipoles, typically 50–300 m long, which are often approximated as point dipoles (Streich & Becken, 2011). The Coulomb-gauged primary potentials for a horizontal electric dipole have been derived from the Lorentz-gauged potentials by Liu *et al.* (2010). For the case of a homogeneous medium described by an uniform electric conductivity, σ_p, they are of the form:

$$A_{px} = \frac{Id\mu_0}{4\pi}\int_0^\infty \frac{\lambda}{\alpha_0}e^{-\alpha_0|z-z_s|}J_0(\lambda\rho)\mathrm{d}\lambda + \frac{Id}{4i\pi\sigma_p\omega}\int_0^\infty \frac{\lambda}{\alpha_0}e^{-\alpha_0|z-z_s|}\frac{\partial^2}{\partial x^2}J_0(\lambda\rho)\mathrm{d}\lambda, \tag{2.8}$$

$$A_{py} = \frac{Id}{4i\pi\sigma_p\omega}\int_0^\infty \frac{\lambda}{\alpha_0}e^{-\alpha_0|z-z_s|}\frac{\partial^2}{\partial x\partial y}J_0(\lambda\rho)\mathrm{d}\lambda, \tag{2.9}$$

$$A_{pz} = \frac{Id}{4i\pi\sigma_p\omega}\int_0^\infty -\mathrm{sign}(z-z_s)\lambda e^{-\alpha_0|z-z_s|}\frac{\partial}{\partial x}J_0(\lambda\rho)\mathrm{d}\lambda, \tag{2.10}$$

$$\Psi_p = 0, \tag{2.11}$$

where $a_{0j} = \sqrt{\lambda^2 - k_{0j}^2}$ and $\rho = \sqrt{x^2 + y^2}$.

Slowly-convergent integrals of this type are usually calculated using Hankel transform filters (Kong, 2007). Integrals 2.8 – 2.10 can be transformed into the so-called closed form using the Sommerfeld identity and derivative relationships (see e.g. Ward & Hohmann, 1988; Chave & Cox, 1982):

$$\int_0^\infty \frac{\lambda}{\alpha_0} e^{-\alpha_0|z-z_s|} J_0(\lambda\rho)\mathrm{d}\lambda = \frac{e^{ik_0R}}{R}, \tag{2.12}$$

$$\int_0^\infty \frac{\lambda}{\alpha_0} e^{-\alpha_0|z-z_s|} \frac{\partial^2}{\partial x^2} J_0(\lambda\rho)\mathrm{d}\lambda = \frac{\partial^2}{\partial x^2}\left(\frac{e^{ik_0R}}{R}\right), \tag{2.13}$$

$$\int_0^\infty \frac{\lambda}{\alpha_0} e^{-\alpha_0|z-z_s|} \frac{\partial^2}{\partial x \partial y} J_0(\lambda\rho)\mathrm{d}\lambda = \frac{\partial^2}{\partial x \partial y}\left(\frac{e^{ik_0R}}{R}\right), \tag{2.14}$$

$$\int_0^\infty \lambda e^{-\alpha_0|z-z_s|} \frac{\partial}{\partial x} J_0(\lambda\rho)\mathrm{d}\lambda = \frac{\partial}{\partial x}\left(\frac{|z-z_s|e^{ik_0R}}{R^3}(1ik_0R)\right). \tag{2.15}$$

By applying these expressions to integrals 2.8 – 2.10, the analytical expressions for the primary potentials in the closed form has been derived:

$$A_{px} = \frac{Id\mu_0}{4\pi}\frac{e^{ik_{0x}R}}{R} - \frac{Id}{4i\pi\sigma_{px}\omega}\frac{e^{ik_{0x}R}}{R^5}(x^2(k_{0x}^2R^2 + 3ik_{0x}R - 3) + R^2ik_{0x}R^3), \tag{2.16}$$

$$A_{py} = -\frac{Id}{4i\pi\sigma_{py}\omega}\frac{xye^{ik_{0y}R}}{R^5}(k_{0y}^2R^2 + 3ik_{0y}R3), \tag{2.17}$$

$$A_{pz} = -\frac{Id}{4i\pi\sigma_{pz}\omega}\frac{x(z-z_s)e^{ik_{0z}R}}{R^5}(k_{0z}^2R^2 + 3ik_{0z}R3), \tag{2.18}$$

$$\Psi_p = 0, \tag{2.19}$$

where $R = \sqrt{x^2 + y^2 + (z-z_s)^2}$, $k_{0j}^2 = i\omega\mu_0\sigma_{0j}$, I is current intensity and d is the dipole length.

In more general cases, the calculation of primary potentials requires the calculation of Hankel transforms, which is computationally expensive. The more efficient way to do it is to use Hankel transform filters. However, one should keep in mind that these filters have some limitations on the distance, though high-performance ones can evaluate fields very well within source-receiver offsets typically used in CSEM (Kong, 2007). Formulas 2.16 – 2.18 have been tested for different values of ρ, up to 20 skin depths, and proved to be precise.

The point source approximation may not represent a real source with necessary precision at short distances. To obtain EM fields for a finite-length dipole, equations 2.8 – 2.10 should be

integrated along its length. However, while the actual source geometry is crucial in land CSEM surveys that use kilometre-long source wires, in marine surveys the wire geometry has a small impact on the responses (Streich & Becken, 2011). For these reasons, in this work, dipoles are approximated as point sources and the values of $\mathbf{A}_p$ in each node of the mesh are calculated using expressions 2.16 – 2.18.

Governing Equations

Finally, the governing equations, which have to be solved numerically, become:

$$\nabla^2 \mathbf{A}_s + i\omega\mu_0\tilde{\sigma}(\mathbf{A}_s + \nabla\Psi_s) = -i\omega\mu_0\Delta\tilde{\sigma}(\mathbf{A}_p + \nabla\Psi_p), \tag{2.20}$$

$$\nabla \cdot [i\omega\mu_0\tilde{\sigma}(\mathbf{A}_s + \nabla\Psi_s)] = -\nabla \cdot [i\omega\mu_0\Delta\tilde{\sigma}(\mathbf{A}_p + \nabla\Psi_p)], \tag{2.21}$$

where $\Delta\tilde{\sigma} = \tilde{\sigma} - \sigma_p$ is the difference between the conductivity distribution, $\tilde{\sigma}(\mathbf{r})$, whose response needs to be calculated and the background conductivity, σ_p, whose response is already known. Considering that formations in marine environments are typically anisotropic and that ignoring this fact severely affects CSEM modelling and inversion results, I consider an anisotropic conductivity model. In particular, my code assumes transverse anisotropy, which corresponds to many situations encountered in actual geologic basins (Newman *et al.*, 2010) and for which tensor $\tilde{\sigma}$ has the following form:

$$\tilde{\sigma} = \begin{vmatrix} \sigma_h & 0 & 0 \\ 0 & \sigma_h & 0 \\ 0 & 0 & \sigma_v \end{vmatrix}, \tag{2.22}$$

where $\sigma_x = \sigma_y = \sigma_h$ is the horizontal conductivity and $\sigma_z = \sigma_v$ denotes the vertical conductivity. The presented approach can be also used in the case of generalised anisotropy when the tensor has six independent elements.

Boundary Conditions

The boundaries of a domain are assumed to be located far away from the transmitter – at a distance where the EM field has negligible values. Therefore, for the secondary EM potentials, homogeneous Dirichlet boundary conditions have been imposed on the outer boundary of the domain, Γ:

$$(\mathbf{A}_s, \Psi_s) = (\mathbf{0}, 0) \text{ on } \Gamma. \tag{2.23}$$

Equations (2.20) and (2.21) together with the boundary condition (2.23) fully describe EM induction caused by dipole or current-loop sources in anisotropic heterogeneous electrically conductive media.

2.2 Finite-Element Analysis

The finite-element method (Burnett, 1987) is a computer-aided mathematical technique for obtaining approximate numerical solutions to the abstract equations of calculus that predict the response of physical systems subjected to external influences. The mathematical structure of the FE approach identifies three principal operations that are present in every FE analysis: construction of a trial solution, application of an optimising criterion and estimation of accuracy.

A physical problem to be solved has its corresponding mathematical formulation in terms of differential equations and boundary conditions. This formulation contains an unknown function, denoted by $X(x, y, z)$ for 3-D problems. For most practical problems it is impossible to determine the exact solution to these equations, i.e. to find an explicit expression for X, in terms of known functions, which exactly satisfies the governing equations and boundary conditions. As an alternative, the FE method seeks an approximate solution, i.e. an explicit expression for X, in terms of known functions, which only approximately satisfies the governing equations and boundary conditions. Such an approximate solution is denoted by the letter X with a tilde over it. Thus, $\tilde{X}$ denotes an approximate solution, whereas X denotes the exact solution.

The FE technique obtains an approximate solution by using the classical trial-solution procedure. This procedure forms the basic structure of every FE analysis. The trial-solution procedure is characterised by three principal operations. These are, in the order of application:

1. Construction of a trial solution for $\tilde{X}$,
2. Application of an optimising criterion to $\tilde{X}$,
3. Estimation of the accuracy of $\tilde{X}$.

Construction of a Trial Solution

The first operation involves the construction of a trial solution $\tilde{X}(x, y, z; \mathbf{x})$ in the form of a finite sum of functions:

$$\tilde{X}(x, y, z; \mathbf{x}) = \phi_0(x, y, z) + x_1\phi_1(x, y, z) + \cdots + x_N\phi_N(x, y, z), \tag{2.24}$$

where x, y, z represent all the independent variables in the problem. $\phi_0(x, y, z), \phi_1(x, y, z), \ldots,$ $\phi_N(x, y, z)$ are known functions called trial (basis or shape) functions. Coefficients $x_1, x_2, \ldots, x_N$ are unknown parameters frequently called degrees of freedom (DOF) or, sometimes, generalised coordinates. So, it is usual to say that $\tilde{X}(x, y, z; \mathbf{x})$ has N DOFs.

The construction of a trial solution consists of building expressions for each of the trial functions in terms of specific, known functions. From a practical standpoint, it is important

to use functions that are algebraically as simple as possible and also easy to work with – for example, polynomials. One of the main attractions of FM is that it provides a systematic procedure for constructing trial functions and the procedure can be automated on a computer. Indeed, the very essence of the method lies in the special manner in which the trial functions are constructed. Once that specific expressions for each trial function $\phi_i(x, y, z)$ have been established, only parameters x_i remain undetermined.

Application of an Optimising Criterion

The purpose of the optimising criterion is to determine specific numerical values for each of parameters $x_1, x_2, \ldots, x_N$. A particular set of values for all x_i uniquely defines a particular solution, because then all x_i and $\phi_i(x, y, z)$ in (2.24) are uniquely determined. Since each x_i can assume an infinity of possible values ($-\infty < x_i < +\infty$), there is an N-fold infinity of possible solutions. It is the job of the optimising criterion to select from all these possibilities the best (or optimum) solution, i.e. the best set of values for x_i.

There are two types of optimising criteria that have played a dominant role historically as well as in FE:

1. Methods of weighted residuals (MWR), which are applicable when the governing equations are differential equations,
2. The Ritz variational method (RVM), which is applicable when the governing equations are variational (integral) equations.

Methods of weighted residuals seek to minimise an expression of error in the differential equation (not the unknown function itself). There are many different MWR criteria and the four most popular ones are:

1. The collocation method,
2. The sub-domain method,
3. The least-squares method,
4. The Galerkin method.

When talking about MWR, a quantity called residual must be defined. Namely, if all terms of the differential equation are transferred to the left-hand side, the right-hand side (RHS) will be zero. This means that if the exact solution were substituted for $X(x, y, z)$ on the left-hand side, then the RHS would be identically zero over the entire domain. If any other function, such

as an approximate trial solution $\tilde{X}(x, y, z; \mathbf{x})$, were substituted for $X(x, y, z)$, the result would be a non-zero function called the residual of the equation, denoted by $R(x, y, z; \mathbf{x})$.

The central idea of all the MWR criteria is: find numerical values for $x_1, x_2, \ldots, x_N$ which will make $R(x, y, z; \mathbf{x})$ as close to zero as possible for all the values of (x, y, z) throughout the entire domain. Namely, the exact solution, by definition, is the function that satisfies the differential equation over the entire domain and the boundary conditions on the boundary. Any function that satisfies the differential equation over the entire domain must also make the residual zero over the entire domain, and vice versa. If it is possible to find $\tilde{X}(x, y, z; \mathbf{x})$ that makes $R(x, y, z; \mathbf{x}) = 0$ everywhere in the domain, and if the boundary conditions are also satisfied exactly, then $\tilde{X}(x, y, z)$ must be the exact solution, $X(x, y, z)$. This conclusion is valid for any reasonably well-posed problem for which exists only one exact solution. Therefore, if a particular $\tilde{X}(x, y, z; \mathbf{x})$ makes $R(x, y, z; \mathbf{x})$ deviate only slightly form zero, then $\tilde{X}(x, y, z)$ is probably very close to $X(x, y, z)$.

Application of an MWR criterion produces a set of algebraic equations and their solution is the best set of numerical values for x_i. This means that the original differential equation is transformed into an approximately equivalent system of algebraic equations and that the problem is converted from its calculus formulation to an algebraic formulation which is much easier to solve. Each different MWR criterion will determine a different set of values resulting in many different approximate solutions. However, in almost all FE applications, the Galerkin method is used. In this method, for each parameter x_i, it is required that a weighted average of $R(x, y, z; \mathbf{x})$ is zero over the entire domain. The weighting functions are the trial functions $\phi_i(x, y, z)$ associated with each x_i. In the end, a trial solution with N parameters yields a system of N residual equations:

$$\iiint_\Omega R(x, y, z; \mathbf{x})\phi_1(x, y, z)\,\mathrm{d}v = 0, \tag{2.25}$$

$$\iiint_\Omega R(x, y, z; \mathbf{x})\phi_2(x, y, z)\,\mathrm{d}v = 0, \tag{2.26}$$

$$\vdots$$

$$\iiint_\Omega R(x, y, z; \mathbf{x})\phi_N(x, y, z)\,\mathrm{d}v = 0. \tag{2.27}$$

Variational principles (sometimes referred to as extremum or minimum principles) seek to minimise, or find an extremum in, some physical quantity, such as energy. The most popular method of this kind is the Ritz variational method.

By applying one of these criteria to the trial solution $\tilde{X}(x, y, z; \mathbf{x})$, the best set of values for all x_i is determined, which means that the best solution is obtained. This solution is called an approximate solution since hopefully, and usually, it is a reasonable approximation to the exact solution. However, since $\tilde{X}(x, y, z)$ is only approximate solution, there are questions of accuracy that naturally arise.

Estimation of the Accuracy

Without some indication of the closeness of the approximate solution $\tilde{X}(x, y, z)$ to the exact solution $X(x, y, z)$, the solution is effectively worthless. The closeness may be expressed by the error $E(x, y, z)$, which is simply the difference between $X(x, y, z)$ and $\tilde{X}(x, y, z)$:

$$E(x, y, z) = X(x, y, z) - \tilde{X}(x, y, z). \tag{2.28}$$

$E(x, y, z)$ is also referred to as the pointwise error because it expresses the error at each point (x, y, z) in the domain. However, equation (2.28) cannot be used to calculate $E(x, y, z)$ since it contains the exact solution which is generally unknown. In fact, it is impossible to calculate $E(x, y, z)$ exactly in an actual numerical problem. If it would be possible, then it would be merely enough to add $E(x, y, z)$ to $\tilde{X}(x, y, z)$ to get $X(x, y, z)$, in which case the error would be zero. Therefore, $E(x, y, z)$ should be estimated in some way. And there are some practical ways to do that.

If after an error estimation it is found out that the magnitude of the estimated error is too large to be acceptable, the error has to be decreased. Namely, we have to return to the first operation and construct a different trial solution that contains more DOFs than the first one. One way to do this is to add a few more trial functions (and hence DOFs) to the previous trial solution. Repeating the second and third operations will then generate a second approximate solution which hopefully will yield a lower error estimate. If the new estimate is still unacceptable, then the cycle can be repeated again and again with successively improved trial solutions (more and more DOFs) until an acceptable error estimate is obtained.

In order to develop a node-based FE code that solves the 3-D CSEM induction problem described previously in Section 2.1, I have followed the so-called '12-step procedure' (Burnett, 1987), which I have reduced to a slightly shorter version that has 10 steps.

10-step Finite-Element Procedure

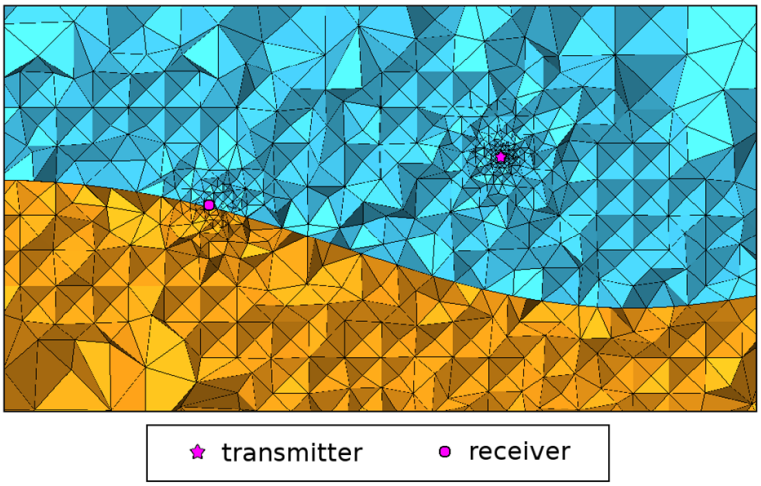
The first five steps involve a theoretical pencil-and-paper manipulation of PDEs. The second five steps involve numerical computations usually performed on a computer. In fact, the steps 6 through 10 describe the basic structure of the node-based FE program.

Preprocessing

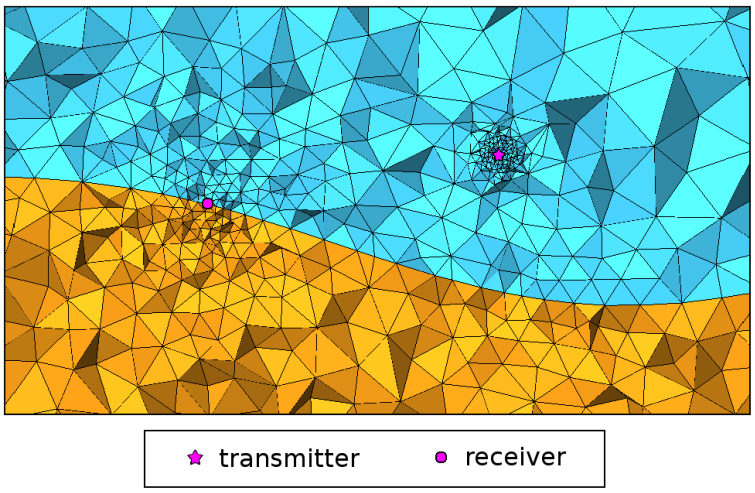
Mesh Generation: At the very beginning of every node-based FE solution of a boundary-value problem, the 3-D problem domain must be discretised into a mesh of non-overlapping polyhedral elements with nodes at the vertices. Adjacent elements touch without overlapping and there are no gaps between the elements. Meshing is very important part of the modelling process since the properties and the quality of a mesh, as well as of its elements, greatly affect numerical features of the FE linear system and, consequently, the convergence of employed iterative methods. Also, the characteristics of the mesh have a considerable impact on the quality and the accuracy of a solution.

In order to carry out the numerical tests, which will be presented later in Chapter 4, I have used ANSYS ICEM CFD mesh-generation software to create the required meshes. Although the program supports different types of elements, which makes it easy to shape very irregular and complex geometries, I have limited myself to linear tetrahedra to build the meshes. ICEM includes both the Octree and the Delaunay tetrahedral mesh-generation techniques (Fig. 2.2). The proper size of the mesh elements is chosen by the skin depth criterion (Commer & Newman, 2008). In FE modelling, it is not necessary to create a fine mesh over the whole domain. It is enough to make local mesh refinements in the regions where field gradients are large, as well as in some parts of the domain in which it is preferable to obtain a solution of higher accuracy. Therefore, the meshes that I have created for the tests have been refined in such places, for example close to the source or receivers. However, at distances of a few skin depths, sizes of tetrahedra can be quite large and keep growing towards the computational boundaries, which are typically located far away to make the boundary effects negligible (Um *et al.*, 2012). A series of experiments have shown that adaptive refinements of a mesh near the regions of interest greatly improve the quality of a solution compared with solutions obtained using simple uniform meshes. Also, the tests have shown that the use of unstructured meshes has the advantage of greatly optimising the number of elements in a mesh without affecting the solution accuracy. On the other hand, unstructured meshes can slow the convergence rate. All this will be demonstrated later in Chapter 4, Subsection 4.1.1.

In order to perform speed-up tests and study solver convergence on large meshes, I have used in this work the mesh multiplication (MM) strategy discussed in Houzeaux *et al.* (2012). The MM scheme consists of subdividing recursively the elements of the original mesh (referred to as the zero-level mesh) in parallel. When using tetrahedra, new nodes are added only on edges and faces and thus the number of elements is multiplied by 8 at each MM level. In the



(a) Slice of a marine CSEM mesh in the X-Z plane generated by the Octree mesh-generation technique.



(b) Slice of a marine CSEM mesh in the X-Z plane generated by the Delaunay mesh-generation technique.

Figure 2.2: Slices of marine CSEM meshes in the X-Z plane generated by ANSYS ICEM CFD mesh-generation software.

previous reference, starting with a tetrahedral mesh that had 30 million elements, a 1.92 billion element mesh was obtained in a few seconds after 2 MM levels.

Theoretical Development

Strong Problem Formulation: At the beginning of the theoretical development, I have decomposed the vector quasi-Helmholtz equation into three scalar equations using the Cartesian coordinate system. This makes the following calculations much easier. Consequently, instead of one vector and one scalar PDE, I have been solving the set of four scalar equations representing the strong formulation of the problem:

$$\nabla^2 A_{sx} + i\omega\mu_0\sigma_x \left(A_{sx} + \frac{\partial \Psi_s}{\partial x} \right) = -i\omega\mu_0 \Delta\sigma_x A_{px}, \tag{2.29}$$

$$\nabla^2 A_{sy} + i\omega\mu_0\sigma_y \left(A_{sy} + \frac{\partial \Psi_s}{\partial y} \right) = -i\omega\mu_0 \Delta\sigma_y A_{py}, \tag{2.30}$$

$$\nabla^2 A_{sz} + i\omega\mu_0\sigma_z \left(A_{sz} + \frac{\partial \Psi_s}{\partial z} \right) = -i\omega\mu_0 \Delta\sigma_z A_{pz}, \tag{2.31}$$

$$i\omega\mu_0 \nabla \cdot [\tilde{\sigma}\mathbf{A}_s] + i\omega\mu_0 \nabla \cdot [\tilde{\sigma}\nabla\Psi_s] = -i\omega\mu_0 \nabla \cdot [\Delta\tilde{\sigma}\mathbf{A}_p]. \tag{2.32}$$

In these equations I have omitted the terms with primary scalar potential since for dipole and loop sources $\Psi_p = 0$.

Step 1: Write the Galerkin residual equations for a typical element.

In order to obtain these equations, I have derived the expression for the residual, $R(x, y, z; \mathbf{x})$, of each governing equation that I have been solving. Having these expressions, I have obtained four sets of n Galerkin residual equations:

$$\int_\Omega R(x, y, z; \mathbf{x}) \phi_i(x, y, z) \, \mathrm{d}v = 0, \qquad i = 1, 2, \ldots, n \tag{2.33}$$

where n is the number of nodes of a typical element. I have chosen a linear tetrahedron as a typical element, so $n = 4$ in this case. $\phi_i(x, y, z)$ are known trial (shape) functions, each of which is associated with the corresponding node i of the element. They are used for the construction of a trial solution of an unknown function within the element. In this case, there are four scalar unknown functions that have to be determined, three Cartesian components of the vector magnetic potential and the scalar electric potential.

Step 2: Integrate by parts.

In this step, I have transformed the integrals that contain the highest derivative in equations (2.33). Namely, the four terms in the strong formulation that contain second-order derivatives have been integrated by parts to reduce the order of the differentiation by one. Three integration-by-parts formulas have been used. The first one is the Green's first identity:

$$\int_{\Omega} u\nabla^2 v \,\mathrm{d}v = -\int_{\Omega} \nabla v \cdot \nabla u \,\mathrm{d}v + \oint_{\Gamma} u(\nabla v \cdot \mathbf{n}) \,\mathrm{d}s, \tag{2.34}$$

where $\oint_{\Gamma} u(\nabla v \cdot \mathbf{n}) \,\mathrm{d}s$ is the surface term, Γ is the boundary of Ω region and $\mathbf{n}$ is the outward pointing unit normal of a surface element $\mathrm{d}s$. The second is the identity:

$$\int_{\Omega} u\nabla \cdot [\sigma\nabla v] \,\mathrm{d}v = -\int_{\Omega} \sigma\nabla v \cdot \nabla u \,\mathrm{d}v + \oint_{\Gamma} u(\sigma\nabla v \cdot \mathbf{n}) \,\mathrm{d}s, \tag{2.35}$$

where $\oint_{\Gamma} u(\sigma\nabla v \cdot \mathbf{n}) \,\mathrm{d}s$ is the surface term. The third formula is:

$$\int_{\Omega} u\nabla \cdot [\sigma\mathbf{A}] \,\mathrm{d}v = -\int_{\Omega} \sigma\mathbf{A} \cdot \nabla u \,\mathrm{d}v + \oint_{\Gamma} u(\sigma\mathbf{A} \cdot \mathbf{n}) \,\mathrm{d}s, \tag{2.36}$$

where $\oint_{\Gamma} u(\sigma\mathbf{A} \cdot \mathbf{n}) \,\mathrm{d}s$ is the surface term. It is obvious that this procedure transforms an integral into two new terms: a boundary term (a surface term) and an integral of one lower order. As a result, after substituting these new terms into the original equation, all loading terms, both in the interior and on the boundary of a domain, pass to the right-hand-side (RHS). In this case, the surface terms, which are 2-D integrals over the boundary of the domain, vanish because trial functions ϕ_i are zero on the boundary.

After this step, I have got the so-called 'weak formulation' of the problem:

$$-\int_{\Omega} \nabla\phi_i \cdot \nabla A_{sx} \,\mathrm{d}v + i\omega\mu_0 \int_{\Omega} \sigma_x\phi_i \left(A_{sx} + \frac{\partial\Psi_s}{\partial x}\right) \mathrm{d}v = -i\omega\mu_0 \int_{\Omega} \Delta\sigma_x\phi_i A_{px} \,\mathrm{d}v, \tag{2.37}$$

$$-\int_{\Omega} \nabla\phi_i \cdot \nabla A_{sy} \,\mathrm{d}v + i\omega\mu_0 \int_{\Omega} \sigma_y\phi_i \left(A_{sy} + \frac{\partial\Psi_s}{\partial y}\right) \mathrm{d}v = -i\omega\mu_0 \int_{\Omega} \Delta\sigma_y\phi_i A_{py} \,\mathrm{d}v, \tag{2.38}$$

$$-\int_{\Omega} \nabla\phi_i \cdot \nabla A_{sz} \,\mathrm{d}v + i\omega\mu_0 \int_{\Omega} \sigma_z\phi_i \left(A_{sz} + \frac{\partial\Psi_s}{\partial z}\right) \mathrm{d}v = -i\omega\mu_0 \int_{\Omega} \Delta\sigma_z\phi_i A_{pz} \,\mathrm{d}v, \tag{2.39}$$

$$-i\omega\mu_0 \int_{\Omega} \tilde{\sigma}\nabla\phi_i \cdot \mathbf{A_s} \,\mathrm{d}v - i\omega\mu_0 \int_{\Omega} \tilde{\sigma}\nabla\phi_i \cdot \nabla\Psi_s \,\mathrm{d}v = i\omega\mu_0 \int_{\Omega} \Delta\tilde{\sigma}\nabla\phi_i \cdot \mathbf{A_p} \,\mathrm{d}v. \tag{2.40}$$

The weak solution is unique and satisfies both the strong and the weak formulation of the boundary-value problem. The equivalence of the weak and the strong solution is a fundamental property of Euler-type second-order PDEs. From this point on, my main goal has become to find the weak solution using FE approximations.

Step 3: Substitute general forms of element trial solutions into the interior integrals in the residual equations. The resultant formal expressions are the element equations.

The general forms of the element trial solutions are:

$$\tilde{\mathbf{A}}_s(x,y,z;A_{sx},A_{sy},A_{sz}) = \sum_{i=1}^{\mathrm{n}}(A_{sxi}\phi_i(x,y,z)\vec{x} + A_{syi}\phi_i(x,y,z)\vec{y} + A_{szi}\phi_i(x,y,z)\vec{z}) \tag{2.41}$$

and

$$\tilde{\Psi}_s(x,y,z;\Psi_s) = \sum_{i=1}^{\mathrm{n}} \Psi_{si}\phi_i(x,y,z), \tag{2.42}$$

where $\phi_i(x,y,z)$ are already described linear nodal basis functions of the element, while (A_{sxi}, A_{syi}, A_{szi}, Ψ_{si}) are unknown coefficients in node i of the element that have to be found by the FE analysis. For the element with n nodes, there are $4n$ coefficients, which are the unknown values of the secondary EM potentials in the element nodes. After the substitution of these expressions into the residual equations, I have obtained a set of $4n$ element equations. The abbreviated matrix notation of this set is:

$$\mathbf{A}^{(e)}\mathbf{x}^{(e)} = \mathbf{b}^{(e)}, \tag{2.43}$$

where matrix $\mathbf{A}^{(e)}$ is the element matrix of coefficients that multiply the vector of unknown parameters, $\mathbf{x}^{(e)}$. The matrix is usually referred to as the stiffness matrix. Vector $\mathbf{x}^{(e)}$ contains the unknown secondary EM potentials in the nodes of the element that are stored in the following way:

$$\mathbf{x}^{(e)} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T \qquad \mathbf{x}_i = [A_{sxi}, A_{syi}, A_{szi}, \Psi_{si}]^T. \tag{2.44}$$

The vector of loading terms on the RHS, $\mathbf{b}^{(e)}$, is usually referred to as the load or force vector, since it represents the source contribution to the FE linear system of equations.

The element system of equations has the following matrix form:

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \ldots & \mathbf{A}_{1n} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \ldots & \mathbf{A}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{n1} & \mathbf{A}_{n2} & \ldots & \mathbf{A}_{nn} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_n \end{bmatrix}. \tag{2.45}$$

The element FE matrix is a $4n \times 4n$ complex block matrix composed of 4×4 sub-matrices, $\mathbf{A}_{ij}$ ($i,j = 1,\ldots,n$), of the following form:

$$\mathbf{A}_{ij} = \begin{pmatrix} \gamma_{ij}\mathbf{I}_{33} & i\omega\mu_0 \int_\Omega \tilde{\sigma}\phi_i \nabla\phi_j \,\mathrm{d}v \\ [i\omega\mu_0 \int_\Omega \tilde{\sigma}\nabla\phi_i\phi_j \,\mathrm{d}v]^T & -i\omega\mu_0 \int_\Omega \tilde{\sigma}\nabla\phi_i \cdot \nabla\phi_j \,\mathrm{d}v \end{pmatrix}, \tag{2.46}$$

where $\mathbf{I}_{33}$ is the 3×3 identity matrix and γ_{ij} is the scalar function:

$$\gamma_{ij} = -\int_\Omega \nabla\phi_i \cdot \nabla\phi_j \,\mathrm{d}v + i\omega\mu_0 \int_\Omega \tilde{\sigma}\phi_i\phi_j \,\mathrm{d}v. \tag{2.47}$$

The electrical conductivity is assumed to be constant over each element, which simplifies the calculation of the presented element integrals. The entries of the load vector are of the form:

$$\mathbf{b}_i = -i\omega\mu_0 \left[\sum_k A_{pxk}\zeta_{ikx} \quad \sum_k A_{pyk}\zeta_{iky} \quad \sum_k A_{pzk}\zeta_{ikz} \quad -\sum_k (A_{pxk}\theta_{ikx} + A_{pyk}\theta_{iky} + A_{pzk}\theta_{ikz})\right]^T, \tag{2.48}$$

where

$$\zeta_{ikx} = \int_\Omega \Delta\sigma_x \phi_i \phi_k \, \mathrm{d}v, \qquad \zeta_{iky} = \int_\Omega \Delta\sigma_y \phi_i \phi_k \, \mathrm{d}v, \qquad \zeta_{ikz} = \int_\Omega \Delta\sigma_z \phi_i \phi_k \, \mathrm{d}v,$$

$$\theta_{ikx} = \int_\Omega \Delta\sigma_x \frac{\partial\phi_i}{\partial x} \phi_k \, \mathrm{d}v, \qquad \theta_{iky} = \int_\Omega \Delta\sigma_y \frac{\partial\phi_i}{\partial y} \phi_k \, \mathrm{d}v, \qquad \theta_{ikz} = \int_\Omega \Delta\sigma_z \frac{\partial\phi_i}{\partial z} \phi_k \, \mathrm{d}v.$$

Coefficients $(A_{pxi}, A_{pyi}, A_{pzi})$ are values of the EM primary potentials in the element nodes. These values are calculated in the preprocess using appropriate analytical expressions that depend on the source type.

Step 4: Develop specific expressions for the element trial functions.

Each type of elements has nodal basis functions of a certain specific form. This form is the same for all the shape functions of one element. The trial functions are polynomials, i.e. linear functions, because it is essential that they are algebraically as simple as possible and easy to work with.

Therefore, I have used the 3-D polynomial functions that correspond to a linear tetrahedron (Jin, 2002). These functions have the following property:

$$\phi_i(x, y, z) = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{2.49}$$

and furthermore, $\phi_i(x, y, z)$ vanishes when the observation point is at the surface of the tetrahedron opposite to the jth node. As a result, the inter-element continuity is guaranteed.

Step 5: Substitute the element trial functions into the element equations and transform the integrals into a form appropriate for numerical evaluation.

The integrals have been evaluated numerically using the Gauss quadrature rule of integration.

Numerical Computation

Step 6: Specify numerical data for a particular problem.

In this step, the program has to read geometric data, as well as the physical properties and information about sources for a problem to be solved. Therefore, it is necessary to create some files that contain the sizes, shapes and locations of the elements, the coordinates of the nodes, as well as the assigned numbers to each node and element. This means that in these files both nodes and elements of the mesh are defined. Node definition means specification of the coordinates of each node and assignment of a number to each node. Element definition means specification of the node numbers associated with each element, i.e. construction of a connectivity table that lists node numbers associated with each element. As for the physical properties and the sources, they are given in a separate file together with a desired numerical treatment of the problem. This file should be filled in directly by a user for each specific problem.

Step 7: Evaluate the interior terms in the element equations for each element and assemble the terms into the system equations.

In this step, the program numerically evaluates all the interior (non-boundary) terms in the element equations for each element and then assembles those sets of element equations to form the system equations. Numerical evaluation is performed in the following way: for each element, substitute the actual node numbers from the connectivity table into the element equations and then substitute the coordinate values. The next step is to apply one of the two inter-element boundary conditions (IBCs), the so-called 'essential IBC', which ensure continuity of the unknown functions. The only way to apply this condition is to constrain the trial functions themselves. This leads to the constraint equations, of the general form $\mathbf{x}_i = \mathbf{x}_j$, which considerably simplify the system equations. Thanks to them, some of the columns of the stiffness matrix and corresponding equations can be combined (added) leading to the assembled system equations. Namely, if a constraint equation is $\mathbf{x}_i = \mathbf{x}_j$, then rows i and j and columns i and j in the stiffness matrix, as well as rows i and j in the RHS vector, are simply combined, i.e. added. This procedure of applying the essential IBC to the element equations is called assembly. Stated more formally, assembly is the enforcement of continuity between the element trial solutions.

The resultant system equations have the stiffness matrix, $\mathbf{A}$, in which most of the terms are zero, which means that the matrix is sparse. This is a direct consequence of the fact that each of the assembled trial functions is non-zero over only one or two elements and zero over all the remaining elements (the sparsity of matrix $\mathbf{A}$ is determined by the mesh node connectivity – the

integrals in (2.46) are zero if node i is not connected to node j through an edge). The sparsness of the stiffness matrix is an important characteristic of the FE method making it feasible and economical to solve very large problems involving tens, and even hundreds, of thousands of equations.

If there are N nodes in the interior of the mesh, the FE system matrix, $\mathbf{A}$, obtained at this point is a symmetric $4N \times 4N$ sparse complex block matrix composed of 4×4 sub-matrices, $\mathbf{A}_{ij}$ $(i, j = 1, \ldots, N)$, of the form (2.46).

Step 8: Apply the boundary conditions, including the natural inter-element boundary conditions, to the system equations.

In this step, the program applies the boundary conditions and the natural IBCs to the assembled system equations. The natural IBCs express continuity of the flux across the inter-element boundary.

Step 9: Solve the system equations.

Considering that, in this case, the matrix of the resultant linear system is large, sparse, complex and non-Hermitian, I have made a selection of appropriate iterative techniques that can handle a system with such characteristics. I have implemented three different right-preconditioned Krylov subspace methods to solve the system: the biconjugate gradient stabilised (BiCGStab) (Van der Vorst, 1992), quasiminimal residual (QMR) (Freund & Nachtigal, 1991) and generalised minimal residual (GMRES) (Saad & Schultz, 1986). The condition number of $\mathbf{A}$ strongly depends on the particular problem with many factors affecting it, for example high conductivity contrasts in the model (especially the presence of air) or big element size ratios in unstructured grids. The solution of the system is discussed in more detail in the following Section 2.3.

Step 10: Postprocess the solution and estimate its accuracy.

The presented FE code computes the secondary EM potentials, $(\mathbf{A}_s, \Psi_s)$, from which the physically significant vector EM field functions have to be derived, $(\mathbf{E}_s, \mathbf{H}_s)$. To do this, it is necessary to perform a numerical differentiation. I have followed the approach of Badea *et al.* (2001) and have implemented the Moving Least Squares (MLS) interpolation scheme, which allows numerical differentiation of the FE-computed potentials. As the weighting function, the Gaussian function (Alexa *et al.*, 2003) has been chosen:

$$\theta(d) = e^{-\frac{d^2}{h^2}}. \tag{2.50}$$

In order to validate the MLS interpolation and the calculation of the primary potentials, the result of the numerical differentiation has been compared with the fields computed by the analytical formulas (Ward & Hohmann, 1988). For the tests described later, spatial derivatives of the EM potentials were obtained from their FE-computed values at $n = 50$ nearest nodes to a test point. This parameter, as well as h in the weighting function, controls the smoothness of the result, choosing between a local approximation and smoothing out sharp features for a more global approximation. Both parameters have been determined empirically.

2.3 Iterative Solvers

As already described, the discretisation of a partial differential equation leads to a very sparse and usually extremely large system of linear algebraic equations:

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{2.51}$$

This system has a huge, but finite, number of unknowns and it approximates the partial differential equation. It is solved using some algebraic solver that is chosen taking into account characteristics of the system matrix. There are two big groups of methods for solving linear algebraic systems (Saad, 2003):

1. Direct methods are factorisation methods derived from Gauss elimination. In these techniques, the initial matrix $\mathbf{A}$ is transformed by the method, so that, in the end, a completely different matrix is obtained. The main advantage of direct solvers is that they are very robust and predictable. However, on the other hand, they have very big memory and computational requirements, which are due to the increase of the number of non-zero elements. Namely, the number of non-zero elements in a factorised matrix (a matrix gained from the sparse matrix) depends on the shape of the initial matrix and can be largely increased compared to the number of non-zeros in the original sparse matrix. Consecutively, direct methods have problems when dealing with large-scale problems. The most efficient ones can successfully manage problems that have up to 10 million unknowns. Taking into account that realistic problems under consideration in this work normally have hundreds of millions of unknowns, it is clear that direct methods are not the right choice for the numerical scheme that has to deal with such enormous problems.

2. Iterative methods are techniques in which the initial matrix $\mathbf{A}$ is not transformed. There are many different methods of this kind, but all of them have in common a main iteration loop and inside it three basic operations:

1. $\mathbf{q} = \mathbf{A}\mathbf{p}$ (matrix-vector multiplication),

2. $< \mathbf{q}, \mathbf{p} >$ (scalar (dot) product of two vectors),

3. $\mathbf{p} = \alpha\mathbf{p} + \beta\mathbf{q}$ (linear combination of vectors).

Iterative methods for solving general large sparse linear systems have been gaining popularity in many areas of scientific computing. Until recently, direct solution methods were often preferred to iterative methods in real applications because of their robustness and predictable behaviour. However, a number of efficient iterative solvers have been discovered and the increased need for solving very large linear systems triggered a noticeable and rapid shift towards iterative techniques in many applications.

Compared to direct methods, iterative techniques have the advantage of low storage requirements. Also, they can be parallelised much more efficiently than direct methods, thanks to which they are able to deal with huge computational demands of large-scale realistic problems. A problem with iterative solvers is that they are not as robust as direct ones and also they are not generic. Namely, while direct methods guarantee that all matrices, except singular ones, can be solved, iterative techniques cannot be applied to all matrices. Moreover, there is not any mathematical rule that can tell us which iterative solver is the best and most efficient choice for our particular problem. It is true that the choice of the solver depends on some characteristics of the system matrix, but this can only narrow down the list of choices. In order to find the most suitable iterative method for a certain problem, it is necessary to implement different options and to test them empirically.

Considering that the problems under consideration in this work are normally enormous (hundreds of millions of unknowns) and that efficiency is imperative, it is clear that iterative methods are much more suitable for solving the resultant linear system within the 3-D CSEM FE solver that is presented here. Currently, the most practical and common group of iterative techniques used in applications are Krylov subspace methods.

2.3.1 Krylov Subspace Methods

Krylov subspace methods (Saad, 2003) are extensively employed for achieving iterative solutions to sparse linear systems arising from discretisation of PDEs in different application areas. These techniques are based on projection processes, both orthogonal and oblique, onto Krylov subspaces, which are subspaces spanned by vectors of the form $p(\mathbf{A})\mathbf{v}$, where p is a polynomial. In short, these techniques approximate $\mathbf{A}^{-1}\mathbf{b}$ by $p(\mathbf{A})\mathbf{b}$, where p is a 'good' polynomial. There are two groups of these methods – methods derived from, or related to, the Arnold orthogonalisation and methods based on Lanczos bi-orthogonalisation.

As already discussed, it has not been clear which Krylov method is the best choice for the problem that I have been solving and therefore it has been necessary to implement several options and to test them in order to chose the most suitable one. The system of the form (2.51) is characterised by the matrix, $\mathbf{A}$, which is of a very large dimension and, in most cases, extremely sparse, as well as by the given RHS vector, $\mathbf{b}$. The properties of the system matrix, $\mathbf{A}$, determined by the discretisation technique, dictate which particular technique should be employed for solving the system. Therefore, since the system matrix in the case under consideration is complex and non-Hermitian (not self-adjoint), I have implemented GMRES, QMR and BiCGStab algorithms, which are probably the most widely used Krylov subspace methods for the solution of this kind of systems.

These three methods differ in storage requirements, number of calculations in each iteration and robustness. GMRES is a well-known Arnoldi-based method proposed by Saad & Schultz (1986). This method generates a non-increasing sequence of residual norms and, consequently, it always guarantees smooth and monotonically decreasing convergence, which, however, is not necessarily the fastest one. Also, it performs only one matrix-vector multiplication in one iteration. The main disadvantage of pure GMRES is its large storage requirement since the method stores all previously-generated Arnoldi vectors. As an alternative, one can use restarted or truncated GMRES (see e.g. Baker *et al.*, 2005). QMR (Freund & Nachtigal, 1991) and BiCGStab (Van der Vorst, 1992) are two different Lanczos-based approaches. These methods have low requirements for storage capacity, which is, in addition, fixed throughout a linear iteration. The number of iterations that QMR and BiCGStab need for convergence can be approximately the same as for GMRES, but each iteration requires two matrix-vector multiplications. Moreover, the original QMR requires transpose matrix-vector multiplications (although a transpose-free modification exists (Freund, 1993)). Those additional calculations make QMR and BiCGStab computationally more demanding compared to GMRES. Also, these methods produce residuals whose norms oscillate – sometimes quite a lot. For more details about the advantages of these methods, as well as considerations on their practical implementations, convergence and breakdown possibilities, the reader is referred to the book of Saad (2003) and the review paper of Simoncini & Szyld (2007).

The results of convergence tests that have been carried out in order to determine which of the implemented solvers is the best choice for the problem under consideration are presented in Chapter 4, Subsection 4.2.

Preconditioning

Although Krylov subspace methods have many advantages, when it comes to very large sparse linear systems, which appear in typical real applications, all of them quite often converge extremely slowly. In general, the convergence of Krylov subspace methods heavily depends on the condition number of the matrix. Matrices with small condition numbers tend to converge rapidly, while those with large ones converge much slower. And real-life linear systems normally have huge condition numbers. In addition, as already said, the main weakness of iterative solvers in general, compared to direct ones, is lack of robustness (Saad, 2003). The solution to these problems lies in preconditioning. A good preconditioning technique can substantially improve both the efficiency and robustness of an iterative method. Moreover, generally, the reliability of iterative solvers, when dealing with various applications, depends much more on the quality of the preconditioner than on the particular Krylov subspace method that is used.

Preconditioning assumes a transformation of the original system (2.51) into a new one, which is called a preconditioned system, by applying some preconditioning matrix, $\mathbf{M}$, in one of three possible ways. The first one is so-called split preconditioning, which leads to a preconditioned system of the following form:

$$\mathbf{M}_1^{-1}\mathbf{A}\mathbf{M}_2^{-1}\mathbf{y} = \mathbf{M}_1^{-1}\mathbf{b}, \qquad \mathbf{y} = \mathbf{M}_2\mathbf{x} \tag{2.52}$$

where $\mathbf{M}_1$ and $\mathbf{M}_2$ are factors of $\mathbf{M}$ ($\mathbf{M} = \mathbf{M}_1\mathbf{M}_2$). If $\mathbf{M}_2 = \mathbf{I}$, preconditioner $\mathbf{M}$ is applied to the left (left preconditioning). And finally, for $\mathbf{M}_1 = \mathbf{I}$, the preconditioner is applied to the right (right preconditioning).

Preconditioning matrix $\mathbf{M}$ can be defined in many different ways but it must satisfy a few critical requirements. First, matrix $\mathbf{M}$ should be close to matrix $\mathbf{A}$ in some sense. More precisely, $\mathbf{M}$ should be a non-singular matrix whose inverse is a good approximation to $\mathbf{A}^{-1}$ ($\mathbf{M}^{-1} \approx \mathbf{A}^{-1}$). Thanks to this, a right-preconditioned matrix, $\mathbf{A}\mathbf{M}^{-1}$, for example, is as close as possible to the identity matrix, $\mathbf{I}$, and therefore the preconditioned system is better conditioned than the original one: $1 \approx \kappa(\mathbf{A}\mathbf{M}^{-1}) \ll \kappa(\mathbf{A})$. Also, matrix $\mathbf{M}$ should be such that linear systems of the form $\mathbf{M}\mathbf{x} = \mathbf{y}$ are inexpensive to solve. Furthermore, $\mathbf{M}$ should be cheap to construct and apply. More details about preconditioning techniques and their applications to iterative methods can be found in (Saad, 2003).

When used for solving systems that appear in 3-D CSEM FE solvers, Krylov methods converge very slowly. The main reasons for the bad convergence are unstructured grids that have big element size ratios and high conductivity contrasts in models (especially if the air layer is included). In order to improve performance of the three chosen Krylov methods, I have implemented their right-preconditioned versions, in which all the calculations are performed in

double complex arithmetic. If instead of the original linear system (2.51), an iterative method needs to solve a right-preconditioned system:

$$\mathbf{A}\mathbf{M}^{-1}\mathbf{M}\mathbf{x} = \mathbf{b} \Longleftrightarrow \mathbf{A}'\mathbf{x}' = \mathbf{b}' \tag{2.53}$$

the algorithm of the solver has to be changed. The change of the system matrix, $\mathbf{A}' = \mathbf{A}\mathbf{M}^{-1}$, affects the operation of matrix-vector multiplication, which is one of the basic operations in every Krylov subspace method. In a right-preconditioned Krylov method, matrix-vector multiplication, $\mathbf{A}\mathbf{p} = \mathbf{q}$, transforms into the following operations:

$$\mathbf{M}^{-1}\mathbf{p} = \mathbf{p}' \qquad \mathbf{A}\mathbf{p}' = \mathbf{q}' \tag{2.54}$$

Also, the new system has a different vector of unknowns, $\mathbf{x}' = \mathbf{M}\mathbf{x}$. Therefore, the final result, $\mathbf{x}$, must be derived from $\mathbf{x}'$:

$$\mathbf{x} = \mathbf{M}^{-1}\mathbf{x}' \tag{2.55}$$

As a preconditioning technique, I have used Jacobi (diagonal) and SSOR preconditioning (Saad, 2003). Very popular preconditioning based on incomplete LU factorisation has not been used because of its rather inefficient parallelisation, which makes it impractical to be incorporated in the fully parallel scheme. The experiments that have been carried out for different Earth models have shown that in many cases, especially more realistic ones, the used preconditioning schemes are not effective enough to ensure convergence to an approximation of desired precision. In order to overcome this problem, I have started a search for a better and more powerful preconditioner that is able to improve the convergence of all three implemented Krylov methods, even in the most challenging cases, and that is also practical for the parallel implementation. During this research, I came across the idea that algebraic multigrid can be very good preconditioner, as already mentioned in Chapter 1, Subsection 1.2.3. After having explored this subject in more depth, I have implemented a more elaborate preconditioning scheme based on algebraic multigrid.

2.4 Algebraic Multigrid

2.4.1 Multigrid

Multigrid is not a single method, not even a group of methods, but a whole approach to solving large and demanding computational problems. There are no ready-to-use multigrid algorithms and recipes. Instead, there are simply concepts and ideas, and some basic strategies, which can lead us and help us to create our own multigrid schemes. Here, I give just a brief overview of

some basic concepts and ideas that are the heart of multilevel methods, and for more information on the topic I refer the reader to Briggs *et al.* (2000) and Trottenberg *et al.* (2001).

The idea of multigrid is based on two principles. The first one is the so-called smoothing principle. Many classical basic iterative methods (relaxation schemes), when applied to discretised elliptic problems, have a strong smoothing effect on the error of any approximation to the exact solution. Namely, during the first several iterations (relaxations), the error decreases rapidly. This is due to the fact that the standard iterations eliminate oscillatory (high-frequency) modes of the error quite efficiently. On the other hand, these iterations are very slow to remove smooth (low-frequency) modes of the error. Therefore, the basic relaxation schemes converge very quickly as long as the error has high-frequency modes, but after removing these modes, the convergence slows down and the entire scheme begins to stall due to the slower elimination of the smooth components. Clearly, the low-frequency components of the error degrade the performance of standard relaxation methods.

The second principle is known as coarse-grid principle. The idea is that any term which is smooth on one grid can be well approximated on some coarser grid (a grid with double the characteristic grid size h, for example) without any essential loss of information. What's more, only low-frequency components on a fine mesh are visible on a coarser one. In addition to this, a smooth wave on a fine grid looks more oscillatory on a coarse grid. Consequently, it can be said that in passing from a fine to a coarse grid, the low-frequency modes become high-frequency ones.

These two principles lead to the following idea: when a relaxation method begins to stall, which means that smooth modes have become dominant in the error, it may be useful to move to a coarser grid and perform the basic iterations on it. Namely, since the smooth modes appear more oscillatory on a coarse mesh, the relaxation scheme can eliminate them more efficiently. In this way, some standard relaxation on different grid levels reduces the corresponding high-frequency components very quickly and, if this process covers all frequencies, the overall error can be eliminated quite rapidly. In addition, any coarse-grid procedure is much less expensive (fewer grid points) than the same procedure on a fine grid.

The described idea has given rise to so-called coarse-grid correction (CGC) strategy, which is the essence of multigrid methods. CGC schemes also incorporate the idea of using the residual equation to iterate on the error directly. The residual equation of the linear system (2.51):

$$\mathbf{A}\mathbf{e} = \mathbf{r}, \tag{2.56}$$

describes a crucial relationship between the error, $\mathbf{e} = \mathbf{x} - \mathbf{x}'$, and the residual, $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}'$ (where $\mathbf{x}'$ is an approximation to the exact solution, $\mathbf{x}$). In addition, it shows that the error

satisfies the same set of equations as the unknown $\mathbf{x}$ when $\mathbf{b}$ is replaced by the residual, $\mathbf{r}$. Taking this into account, it is clear that a relaxation on the original equation (2.51) with an arbitrary initial guess $\mathbf{x}_0$ is equivalent to iterations on the residual equation (2.56) with the specific initial guess $\mathbf{e} = 0$, which makes the idea of CGC valid. The CGC procedure is described in Fig. 2.3.

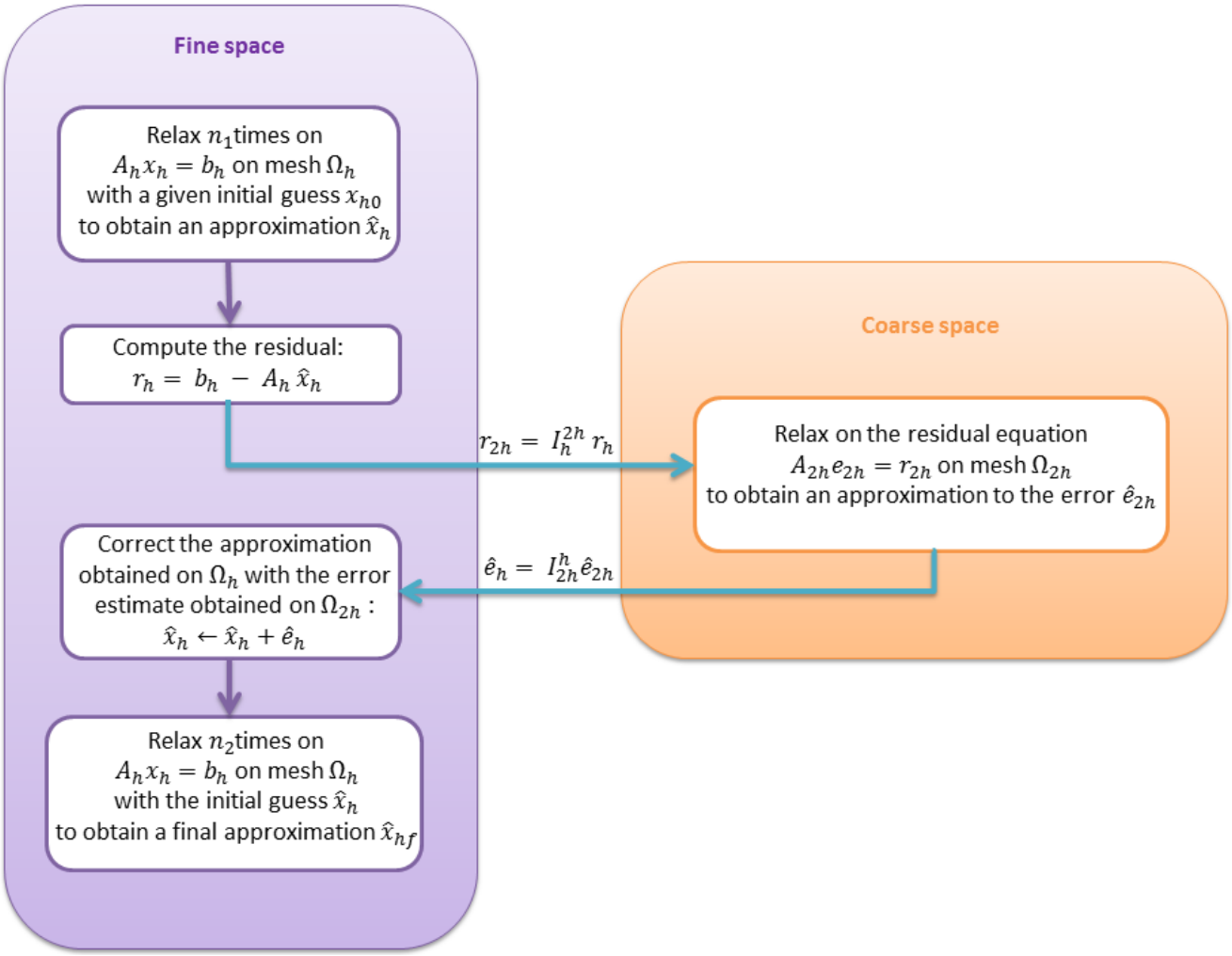


Figure 2.3: Coarse Grid Correction procedure.

The integers n_1 and n_2 are parameters in the scheme that control the number of iterations before and after the coarse grid correction. The given procedure shows that, first, a relaxation method performs n_1 iterations on a fine grid. The idea, as already described, is to let the relaxation method to iterate as long as it is efficient, i.e. until the convergence stalls. In practice, n_1 is often 1, 2 or 3. After n_1 iterations, one has an approximation to the solution that is used to calculate the residual. Since the residual is determined for the fine grid, it has to be transferred to a coarse-grid vector using some restriction operator, I_h^{2h}. Having the coarse-grid residual, it

is possible to solve the residual equation on this coarse grid and obtain the coarse-grid error. According to the procedure, the exact solution of the residual equation on the coarse grid should be obtained. However, if this is not possible, one should approximate the coarse-grid error. When either the exact coarse-grid error or its approximation is determined, the next step is to transfer it to the fine-grid vector by some interpolation operator, I_{2h}^{h}. This fine-grid error is then used to correct the fine-grid approximation that is obtained after n_1 iterations. In the end, the relaxation method performs n_2 additional fine-grid iterations.

A very important feature of this procedure is that its functions are complementary to each other. Namely, the relaxation on the fine grid eliminates the oscillatory components of the error and leaves an error that is relatively smooth. This error is determined by solving the residual equation on the coarse grid and by interpolation of the resultant coarse-grid error. Since the error is smooth, interpolation works very well and the error can be represented accurately on the fine grid. In cases when it is not possible to get the exact solution of the residual equation, it is good idea to approximate it using the relaxation scheme with initial guess equal to zero. This is due to the fact that the error produced on the fine grid, which is quite smooth, appears oscillatory on the coarse grid and therefore is quickly reduced by the relaxation. Having the described CGC procedure as the starting point, one can create a great variety of multilevel methods since each function and element of the procedure can be implemented in numerous different ways.

Another significant characteristic of multigrid techniques is that, unlike in other methods, the number of iterations required to obtain a prescribed accuracy is independent of the mesh size. In this sense, multigrid methods are optimal. On the other hand, a multigrid scheme needs not only the system matrix and the RHS vector, but also a sequence of coarser grids. This makes the implementation of a multigrid technique more challenging than that of some single-grid iterative method. In addition to this, unstructured, irregular grids are especially complicated for multigrid methods. For a given unstructured grid, it is usually not difficult to define a sequence of finer grids, but it may be difficult to define a sequence of reasonable coarser grids that are necessary for multigrid. Therefore, for problems defined on unstructured grids, it is much better to employ algebraic multigrid methods, because these methods construct a hierarchy of coarse grids automatically using only algebraic information contained in the matrix of the resultant linear system.

2.4.2 Algebraic Multigrid as a Solver

Algebraic multigrid (AMG), unlike geometric multigrid, does not require a given problem to be defined on a grid. This is due to the fact that AMG operates directly on the linear sparse

system of algebraic equations. Taking this into account and changing some terminology accordingly, AMG can be described in formally the same way as geometrically based multigrid. Coarse-grid discretisations used in geometric multigrid methods to reduce low-frequency error components correspond to certain systems of equations of reduced dimensions in AMG schemes. However, there are some important conceptual differences. The most significant one is that geometric schemes employ fixed grid hierarchies and hence obtain desirable efficiency by choosing appropriate smoothing processes, while, on the other hand, AMG uses some simple relaxation scheme, such as Jacobi or Gauss-Seidel, as the fix smoother and gains efficiency by selecting suitable coarser levels and transfer operators. In other words, geometric multigrid needs to know a grid hierarchy in advance, while AMG does not have this requirement. In fact, the construction of a hierarchy, which is problem-dependent, is part of an AMG algorithm and is completely automatic. This is possible because the hierarchy construction, which includes the coarsening process itself, the transfer operators as well as the coarse-grid operators, is based completely on algebraic information contained in the given system of equations. Thanks to the fully automatic coarsening process, AMG is quite flexible to adapt to specific requirements of the given problem, as well as very robust in solving large classes of problems despite using very simple smoothers. Furthermore, AMG methods can be designed as black-box solvers, which gives an attractive advantage to this approach. Nonetheless, all these advantages naturally come at a price. Namely, AMG methods distinguish a set-up phase and a solution (cycling) phase. The set-up phase, in which the given problem is analysed and the coarse levels, as well as all the operators, are constructed, has to finish before the solution phase can start. Also, this phase is normally rather costly. Consequently, it introduces significant overhead, which is one of the reasons why AMG is usually less efficient than geometry-based multigrid. However, when it comes to efficiency, AMG should not be regarded as a competitor of geometric multigrid. The strengths of AMG are its robustness, its applicability in cases with complex geometries, which demand unstructured meshes, as well as its applicability to solving problems which cannot be treated using geometric multigrid (non-geometric problems). Therefore, AMG may be considered to be a good alternative to geometric multigrid whenever the latter one is either too difficult to apply or cannot be used at all. Furthermore, AMG can be an efficient alternative to standard numerical methods, such as Krylov subspace methods. In addition to this, as already stated, AMG can be used to create a very efficient preconditioner. Actually, very often, some simplified AMG versions used as preconditioners are better than more complex ones implemented as standalone solvers.

2.5 Algebraic Multigrid Applied as Preconditioning

In this subsection, I present a novel more elaborate preconditioning scheme based on AMG. The AMG preconditioner is based on a very simple coarsening procedure as well as a specific implementation of the one-level coarse-grid correction strategy, which corresponds to $\mathbf{M}^{-1}$ in the standard preconditioning approach that uses some preconditioning matrix, $\mathbf{M}$. Namely, in preconditioners based on AMG, there is no preconditioning matrix $\mathbf{M}$. Therefore, in each matrix-vector multiplication, $\mathbf{A'p} = \mathbf{AM}^{-1}\mathbf{p}$, in order to obtain vector $\mathbf{z} = \mathbf{M}^{-1}\mathbf{p}$, instead of doing multiplication employing explicitly $\mathbf{M}^{-1}$, system $\mathbf{Az} = \mathbf{p}$ is solved using AMG. Considering that AMG, as a complex preconditioning technique, introduces significant overheads in each iteration of the outer Krylov subspace solver, it is important to reduce its cost as much as possible without losing its effectiveness. This is why only one level of CGC is used in this particular preconditioner. Also, the set-up phase is normally extremely expensive part of AMG and therefore it is good idea to implement a simple and fast coarsening algorithm in order to reduce additional costs introduced to the outer iterative solver by AMG preconditioning. In the presented preconditioning scheme, coarsening is based on groups of the mesh nodes that are divided into these groups according to a simple levelisation algorithm based on the spatial distance among the nodes.

The one-level coarse-grid correction (CGC) procedure is implemented in the following way:

- Do p basic relaxations on the original system, $\mathbf{Az} = \mathbf{p}$, with the initial guess $\mathbf{z}_0 = 0$, and get an approximation $\mathbf{z}_p$.
- Find the residual: $\mathbf{r}_p = \mathbf{p} - \mathbf{Az}_p$.
  - Project $\mathbf{r}_p$ on a coarse space: $\tilde{\mathbf{r}}_p = \mathbf{W}^T\mathbf{r}_p$.
    - Solve a coarse-space residual system: $\tilde{\mathbf{A}}\tilde{\mathbf{e}}_p = \tilde{\mathbf{r}}_p$.
  - Project back $\tilde{\mathbf{e}}_p$ on the fine space: $\mathbf{e}_p = \mathbf{W}\tilde{\mathbf{e}}_p$.
- Correct the fine-space approximation: $\mathbf{z}_p = \mathbf{z}_p + \mathbf{e}_p$
- Do p basic relaxations on the original system, $\mathbf{Az} = \mathbf{p}$, with the initial guess $\mathbf{z}_0 = \mathbf{z}_p$, and get the final, i.e. preconditioned, approximation.

As a smoother, it is possible to use one of three basic relaxation methods that I have implemented: Jacobi, Gauss-Seidel and symmetric successive over-relaxation (SSOR). The number of basic iterations $n_1 = n_2 = p$ is usually 1, 2 or 3. For transferring vectors between the fine and a coarse space, projection matrix $\mathbf{W}$ is used, where the restriction operator, I_h^{2h}, is the transpose

of the matrix, $\mathbf{W}^T$, and the interpolation operator, I_{2h}^h, is the matrix itself, $\mathbf{W}$. Matrix $\mathbf{W}$ is also used to build the coarse-system matrix, $\tilde{\mathbf{A}} = \mathbf{W}^T\mathbf{A}\mathbf{W}$. Since the coarse residual system normally has a very small dimension (in the order of 10^2–10^3), a direct method based on LU factorisation is used to solve it.

Considering that the linear system of equations under consideration is mesh-based, the easiest way to define projection matrix $\mathbf{W}$ is by using the points of the mesh. Namely, the nodes of the fine mesh are divided into sub-domains or groups, which are represented by one variable in the coarse space. In theory, $\mathbf{W}$ is a $N \times m$ matrix, where N is the dimension of the fine system and m is the dimension of the coarse system. Each column of $\mathbf{W}$ represents one group of nodes, and the entries in one column are ones for the points that belong to the assigned group and zeroes for all other points. In practice, the matrix $\mathbf{W}$ is not explicitly constructed, as the clustering process is very simple. Let me say that `lgrou(ipoin)` is the array defining to which group, `igrou`, each node, `ipoin`, belongs and that `npoin` is the total number of nodes. Special attention must be paid to the nodes where Dirichlet boundary conditions are applied. One way of dealing with this issue is to assign a null group to these nodes, i.e. `lgrou(ipoin) = 0`. Then, assuming for the sake of clarity that both the fine and the coarse matrices are dense, the construction of the coarse matrix is carried out as follows:

```
A_coarse = 0
do ipoin = 1,npoin
  igrou = lgrou(ipoin)
  if( igrou > 0 ) then
    do jpoin = 1,npoin
      jgrou = lgrou(jpoin)
      if( jgrou > 0 ) then
        A_coarse(igrou,jgrou) = A_coarse(igrou,jgrou) + A_fine(ipoin,jpoin)
      end if
    end do
  end if
end do
```

The restriction, i.e. the projection of a fine vector `x_fine` onto a coarse one `x_coarse`, is performed as:

```
x_coarse = 0
do ipoin=1,npoin
  igrou = lgrou(ipoin)
  if( igrou > 0 ) then
```

```
      x_coarse(igrou) = x_coarse(igrou) + x_fine(ipoin)
    end if
  end do
```

And the inverse operation is given by:

```
  do ipoin=1,npooin
    igrou = lgrou(ipoin)
    if( igrou > 0 ) then
      x_fine(ipoin) = x_coarse(igrou)
    else
      x_fine(ipoin) = 0
    end if
  end do
```

Mesh partitioning into groups of nodes can be performed using **METIS** (Karypis & Kumar, 1995), which is normally employed in practice. Although I have included this possibility in the presented code, I have also implemented the so-called wave-front algorithm to define these groups. Namely, starting from a prescribed point, neighbouring nodes are added into a group until a specified number of points per group is reached. The last set of nodes that are added is used as a starting point for the next group. The procedure is repeated until all points are assigned to some group. In spite of the fact that it is quite simple, this algorithm shows quite good performance, which is in some cases slightly better than that achieved using METIS.

I have designed the presented AMG preconditioner as a black box, so that it can be employed by different iterative methods without any additional modifications of the solver's algorithm. In other words, if there is a preconditioned version of some iterative method, it can use this AMG preconditioner simply by calling a preconditioning subroutine and choosing it among all other available preconditioners.

Chapter 3

Parallel Implementation

In real industrial applications, solution of the three-dimensional electromagnetic (EM) forward problem is computationally extremely demanding – normally, hundreds of millions of EM field unknowns have to be determined. If we take into account that this time-consuming task is a crucial part of any inversion algorithm, it is clear that a highly efficient implementation of 3-D EM modelling is critical for creating a practical interpretation method for 3-D EM data. This desired efficiency can be obtained by parallel computing, which is nowadays widely accepted as a means of handling very large computational tasks. As mentioned in Chapter 2, the only truly efficient 3-D EM modelling code, developed by Alumbaugh *et al.* (1996), is fully parallel. However, this solver is based on the finite-difference (FD) method, which puts some limitations on its applicability. Namely, in situations in which structures that have complex, irregular geometries have to be taken into account, FD cannot guarantee high quality of a solution, which can lead to incorrect interpretations (as explained more elaborately in Chapter 1, Subsection 1.2.3). In order to overcome the limitations of the FD approach, one can employ the finite-element (FE) method. However, despite all the advantages of the FE approach (presented in Chapter 1, Subsection 1.2.3), there are still no fast, parallel FE schemes for 3-D EM modelling. This is the reason why a big portion of my research has been focused on the development of a highly efficient parallel scheme for the 3-D CSEM FE solver presented in this work.

The employed parallelisation strategy is based on the domain decomposition (mesh partitioning) technique using the Message Passing Interface (MPI) programming paradigm for communication among computational units. In addition to this, I have used OpenMP for parallelisation inside of each computational unit. In this way, I have created a powerful hybrid parallel scheme, schematically shown in Fig. 3.1, that accelerates the execution of the forward-problem code to a great extent. The idea is to partition the original problem domain, which normally consists of a huge number of elements, into smaller sub-domains. Thanks to this

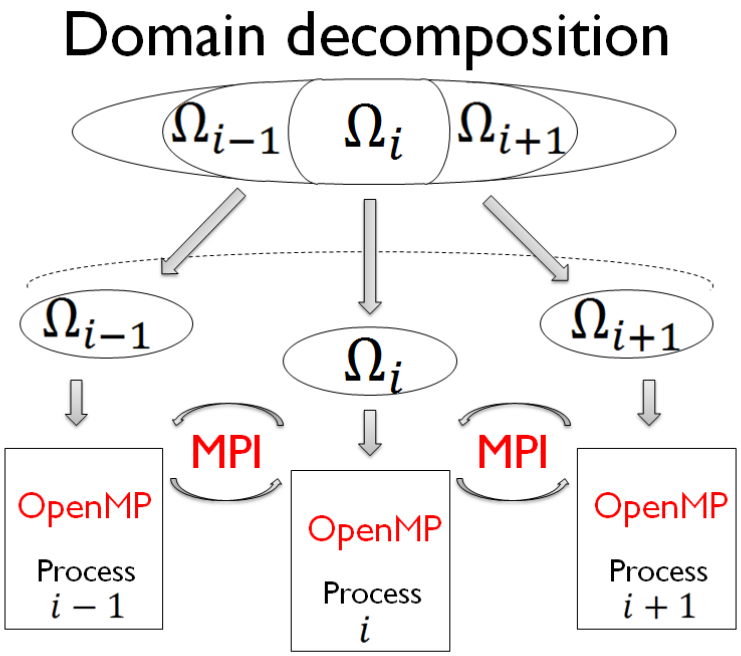


Figure 3.1: Hybrid parallel scheme used for parallelisation of the 3-D CSEM FE solver.

partitioning, many computations can be done simultaneously, i.e. in parallel, which may reduce the total execution time of the program substantially. The mesh partitioning inside the code is performed using **METIS** (Karypis & Kumar, 1995), a set of serial programs for partitioning graphs and FE meshes that can provide very good volume to surface ratios and well-balanced domains for arbitrary meshes, which is the paramount for efficient parallel executions of FE simulations.

In order to perform calculations concurrently, it is necessary to have multiple parallel tasks (processes) – one for each sub-domain. The number of tasks must be specified before the beginning of a program execution. When a system initialises the given number of processes, it assigns to each of them their own unique identifiers. These identifiers, called ranks or task IDs, are sequential integers starting at zero. The process of the rank zero is assigned to be the master while all the other tasks have the role of a slave. The master performs all sequential parts of the code. This means that it reads a mesh and problem parameters, performs partitioning of the mesh into sub-domains, sends the sub-domains and their supplementary data to the corresponding slaves, launches the simulation and writes output files. Slaves, on the other hand, do all time-consuming calculations in parallel. They build element matrices and right-hand-side (RHS) vectors that are then assembled into local system matrices and RHS vectors

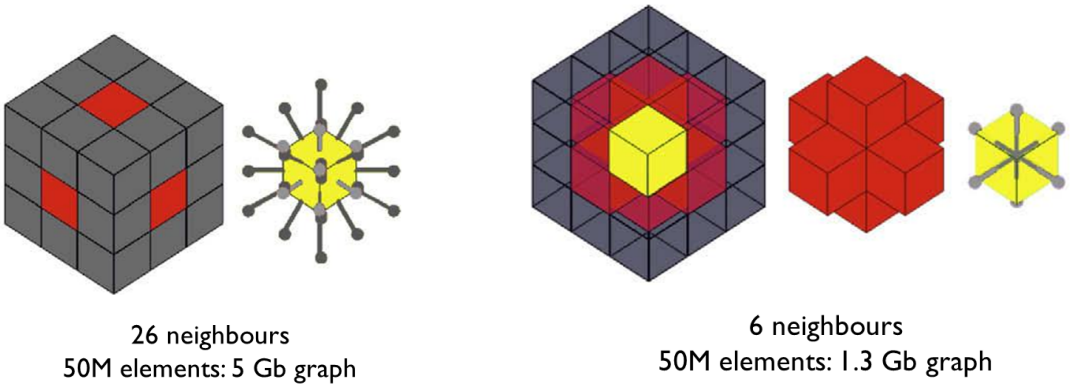
for each sub-domain. These local matrices and RHS vectors are in fact parts of the global system matrix and RHS vector. This means that each slave creates only one part of the global system matrix that corresponds to the sub-domain assigned to it. After this, the slaves solve the resultant global linear system in parallel by executing some iterative method. Since the master does not have any sub-domain assigned to it, it does not perform any computations in the iterative linear solver – it only controls the process and writes in output files after each step.

3.1 Mesh Partitioning

As already mentioned, mesh partitioning is performed using METIS (Karypis & Kumar, 1995). METIS is a family of multilevel partitioning algorithms, i.e. programs for partitioning unstructured graphs and hypergraphs, as well as for computing fill-reducing orderings of sparse matrices. The underlying algorithms used by METIS are based on a state-of-the-art multilevel paradigm that has been shown to produce high quality results and scale to very large problems (**http://glaros.dtc.umn.edu/gkhome/views/metis**).

The main input data for METIS are the element graph of a mesh and weights of the vertices of the graph. For building an element graph, two options are possible. A straightforward option is to consider all the elements that share a node with an element e to be its adjacent elements, i.e. neighbours (Fig. 3.2a). However, such a graph is normally extremely demanding in terms of memory – especially in the case of 3-D meshes. For example, for a mesh of hexahedra that has 50 million elements, the element graph based on node connectivity, which means that every internal element has 26 neighbours, would occupy 5.0 GB (if using four-byte integers). The second strategy, which has shown good load balance results so far, is to take as the adjacent elements to e only the elements that share a face, in the case of 3-D elements, or a side (an edge), in the case of a 2-D mesh, with the element e (Fig. 3.2b). This strategy requires much less memory – for the 50-million-hexahedra example, every inner element has 6 neighbours and thus the element graph occupies 1.3 GB. Considering that meshes used in practice are generally huge (thousands of millions of elements), it is the strategy based on face connectivity that is used in this work. As for the vertex weights, there are also two options. Weights of the vertices of the graph, where a vertex in the graph represents an element in the mesh, can be equal to either 1 or the number of Gauss points of the corresponding element.

As previously said, mesh partitioning is performed by the master. The master is the one who reads data that describe the mesh (geometric information on the elements and nodes of the mesh placed in files automatically created after mesh generation). Having this information, the master is able to compute the element graph of the mesh using node or face connectivity. After



(a) Adjacency based on node sharing.

(b) Adjacency based on face sharing.

Figure 3.2: Two possible strategies for creating the element graph of a 3-D mesh.

creating the graph, it computes weights of the vertices of the graph. In this way, it obtains all the necessary input data for METIS. Having this, it performs graph partitioning by calling a proper METIS function. The output of METIS is simply an array saying for each element to which sub-domain it belongs. This, however, is not enough information for the slaves to perform their parts of the job, let alone to do it efficiently. Therefore, the master has to do some additional computations and to create some new data structures which will be sent to the slaves together with the sub-domains themselves. So, after getting the output array from METIS, the master performs the following operations:

Determine a communication strategy

Communication scheduling is an extremely important aspect that affects efficiency of communication among slaves. Fig. 3.3 illustrates the significance of communication scheduling on a simple example. In this example, each sub-domain has to communicate with all others. In the upper part of the figure, we can observe the consequences of a bad communication strategy, for which communication is carried out in five steps. For example, during the first step, sub-domain 2 cannot communicate with sub-domain 3 until the latter has finished communication with sub-domain 1. In order to optimise communication, which means to reduce the number of communication steps as much as possible, it is necessary that every sub-domain is able to communicate in each communication step. The bottom part of the figure illustrates the optimum scheduling for the given example. Applying this strategy, communication is performed in three steps.

In this work, communication scheduling is done by the master using a colouring strategy. The idea is to schedule communication by colouring the edges of the sub-domain graph (a graph

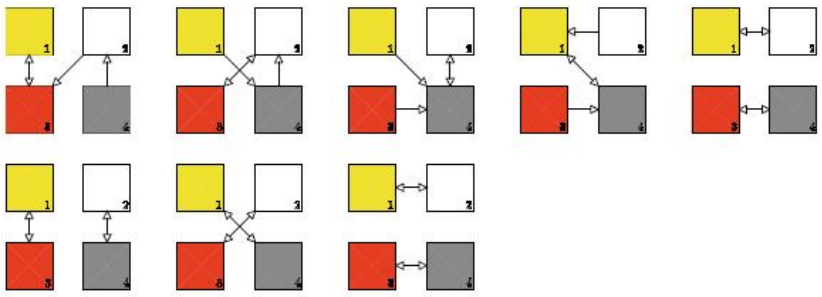


Figure 3.3: Simple illustration of communication scheduling. (Top) Bad communication scheduling in five steps. (Bot.) Optimum communication scheduling in three steps.

in which neighbouring sub-domains are those that communicate among themselves) in such a way that, in the end, communication is performed in as few steps as possible. The strategy is implemented in the following way.

First, the master creates a half-matrix that shows which are neighbouring sub-domains (sub-domains that share some mesh nodes). This matrix, `neighDom(i,j)`, which has only the bottom half and the main diagonal, stores for `i` $\neq$ `j`, i.e. in the bottom half, the number of mesh nodes shared by sub-domains `i` and `j` and for `i` = `j`, i.e. on the main diagonal, the total number of shared (interface) mesh nodes of sub-domain `i`. Using this matrix, an additional array is created. This array, `lneig_par(i)`, stores the number of neighbours of sub-domain `i`. Furthermore, a sub-domain interconnection graph is built using the half-matrix. This graph is stored in two arrays: `adjDom`, which is an adjacency array, and `xadjDom(i)`, an array of pointers that show for each sub-domain where the list of its neighbours starts in vector `adjDom`.

The next step is to create a dual graph for the sub-domain interconnection graph. The dual graph is obtained from the original one in the following way: the edges of the original graph, which represent connections between neighbouring sub-domains, become nodes in the new graph; these nodes in the new graph are connected if they share a node in the original graph, i.e. if they present different connections of the same sub-domain.

After the dual graph is created, the master applies a colouring algorithm to it. Namely, it assigns a colour to every graph node, using the minimal possible number of colours, in such a way that none of the nodes has the same colour as its neighbouring nodes.

Having done colouring, a communication table called `lcomm_par(i,j)` is constructed. The idea of the colouring strategy is that the dual-graph nodes with the same colour represent communications that can be done at the same time. Therefore, less colours mean fewer communication steps. Also, this is why neighbouring nodes cannot have the same colour. Namely, they present different connections of the same sub-domain and hence cannot be performed at

the same time. The communication table contains the information on which sub-domains are connected with communication for each colour, i.e. which sub-domains can communicate at the same time. In ideal case, there are very few colours, i.e. communication steps, and every sub-domain is able to communicate in each of these steps.

Create additional data structures

Starting from the output array of METIS, the master makes some additional data structures that will be sent to the slaves in order to provide them with all the necessary information for their share of the work.

First, the master creates `lnpar_par(i)` array, where `i` represents a node of the mesh. This array stores either the number of a sub-domain, if `i` is an internal node of that sub-domain, or zero, if `i` is a shared (interface) node. At first, this array tells which mesh nodes are internal ones, specifying to which sub-domain each of them belongs, and which are shared nodes. The next step is the distribution of interface nodes between adjacent sub-domains, after which array `lnpar_par(i)` is able to say to which sub-domain each shared node belongs. This means that for each interface node, instead of zero, the negative number of the sub-domain to which the node is assigned is placed in the array. Also, `npoin_par(i)` array is created. This array has the size of the number of partitions and stores the total number of nodes that each partition has. This number includes all the internal nodes that a partition has, as well as all the interface nodes that the partition shares with all its neighbouring sub-domains – not only those interface nodes assigned to it, called own interface nodes, but also those assigned to all its neighbours, called others' interface nodes. This means that every interface node will appear in all sub-domains that share it. Furthermore, `npoin_total`, which is the sum of total numbers of nodes of all sub-domains, is calculated.

Having these data structures, the master is able to separate internal and interface nodes for each sub-domain (here, interface nodes are just those assigned to a sub-domain, i.e. own interface nodes). So, for each sub-domain, it creates permutation and inverse vectors for internal nodes, `permI(i)` and `invpI(i)`, as well as for interface nodes, `permB(i)` and `invpB(i)`. The permutation vectors are of size which is equal to the total number of nodes in the original mesh, but for each sub-domain only those places that belong to its internal/interface nodes are filled with temporary local numbers (numbers in the sub-mesh) of these nodes. The inverse vectors of a sub-domain are of size that equals the number of internal/interface nodes in the sub-domain and they store global numbers (numbers in the original mesh) of these nodes. Also, `nbNodInter`, i.e. the number of internal nodes in a sub-domain, and `nbNodBound`, i.e. the number of interface nodes assigned to the sub-domain, are calculated.

Furthermore, for each sub-domain, the master creates a sub-graph which describes adjacency for internal nodes. These sub-graphs are used as inputs for a METIS function that renumbers the interior nodes of a sub-domain in order to reduce cache misses during gathering operations. For example, when an element system is calculated, the nodes of the corresponding element are observed under their local numbering within the element. However, there is a necessity for some 'sub-global' information on the nodes ('sub-global' means at the level of a sub-mesh). That is why it is needed to perform a gathering operation in order to connect the local (element-level) numbers of the nodes with their 'sub-global' (sub-mesh-level) numbers and to obtain necessary data from 'sub-global' data structures. Also, gathering is crucial in system matrix assembly, when the entries of an element matrix have to be placed in the system matrix of the corresponding sub-domain. Node renumbering helps in such a way that, after it, neighbouring mesh nodes are closer to each other in 'sub-global' data structures, so a cache miss rate in accessing those structures becomes considerably lower. The output of the METIS function is reordered permutation vector `permR(i)`, which has the size of `nbNodInter`.

Using these structures, two new vectors are created: node permutation vector `lnper_par(i)` and vector of inverse node permutation `lninv_par(i)`, both of length that equals the total number of nodes in the mesh, `npoin`. These vectors are created in the following way:

```
offsetI = 0
offsetB = inter
do ipart = 1, npart_par
  ! Number interior nodes
  do ii = 1, nbNodInter
    kk             = invpI(ii)
    jj             = permR(ii) + offsetI
    lnper_par(kk) = jj
    lninv_par(jj) = kk
  end do
  offsetI = offsetI + nbNodInter
  ! Number interface nodes
  do ii = 1, nbNodBound
    kk             = invpB(ii)
    jj             = ii + offsetB
    lnper_par(kk) = jj
    lninv_par(jj) = kk
  end do
  offsetB = offsetB + nbNodBound
end do
```

Vector `lninv_par(i)`, which is the most important one, contains the global numbers of the internal nodes of sub-domain 1, followed by the global numbers of the internal nodes of sub-domain 2, and so on for all the sub-domains. After all the internal nodes of all sub-domains, the interface nodes assigned to sub-domain 1 begin, then the own interface nodes of sub-domain 2 follow, and so on for all the sub-domains.

Finally, having `lninv_par(i)` array, vectors `lninv_loc(i)` and `xlnin_loc(i)` are created. The first one is a vector of inverse node permutation, but unlike `lninv_par(i)`, which has `npoin` elements, it is of size `npoin_total`. This array places the global numbers of all the nodes of all sub-domains in the following order: first all the nodes of sub-domain 1 are placed, then the nodes of sub-domain 2, and so on. The nodes of each sub-domain are organised in such a way that first go the internal nodes followed by all the interface nodes (own interface nodes and others' interface nodes) of the sub-domain. The other array, `xlnin_loc(i)`, whose number of elements is equal to the number of sub-domains plus one, stores a pointer for each sub-domain which shows where the nodes of that sub-domain begin in `lninv_loc(i)` array. The last element of this array stores a pointer to the location after the last node of the last sub-domain. In order to be able to deal with shared nodes in `lninv_loc(i)` vector, some additional structures are created:

- `slfbo_par(i)`, an array of size that equals the number of sub-domains, stores for each sub-domain a pointer that shows where its own interface which has `nbNodBound` nodes begins.

- `bdom(i)`, an array of size (`npoin_total` − the total number of all internal nodes in all sub-domains), contains the list of sub-domains to which interface nodes belong.

- `bpoin(i)`, an array of size (`npoin_total` − the total number of all internal nodes in all sub-domains), contains the local numbering of interface nodes in each of the sub-domains to which they belong.

As for the mesh elements, there are a few data structures created for them, starting from the METIS output:

- `nelem_par(i)`, an array of size that equals the number of sub-domains, stores the number of elements in every sub-domain.

- `leinv_par(i)`, an array of size that equals the total number of elements in the mesh, is a vector of inverse element permutation.

- `leper_par(i)`, an array of size that equals the total number of elements in the mesh, is an element permutation vector.

Vector `leinv_par(i)`, the most important one, contains for every sub-domain the global numbers of its elements. First, the elements of sub-domain 1 are stored, then the elements of sub-domain 2, and so on. There is `leind_par` array of pointers which point to the beginning of the element list for every sub-domain. Also, there is one more pointer for the first location after the `leinv_par(i)` array's end. The code for creating these arrays is:

```
leind_par(1) = 1
do domai = 1, npart_par
  leind_par(domai+1)  = leind_par(domai) + nelem_par(domai)
end do
do ielem = 1, nelem
  domai             = lepar_par(ielem)  ! The METIS output
  jelem             = leind_par(domai)
  leinv_par(jelem)  = ielem
  leper_par(ielem)  = jelem
  leind_par(domai)  = leind_par(domai) + 1
end do
```

Finally, having prepared all the additional data, the master sends them together with the sub-domains to the corresponding slaves, which then solve the problem in parallel getting exactly the same result like in a sequential execution.

3.2 MPI Communication

3.2.1 Synchronous Communication

Having all the necessary data, slaves can perform their tasks in parallel in the following way:

- System assembly – this part of the code is perfectly parallel, which means that each slave can perform its part of the work completely independently from all other slaves – without needing to communicate with anyone.
  - Each slave computes element matrices and RHS vectors for each element that belongs to the assigned sub-domain.
  - Each slave assembles the element matrices and RHS vectors into the local system matrix and RHS vector of the corresponding sub-domain.

- Iterative linear algebraic solver – in order to perform two out of three basic operations that comprise every iterative method, slaves have to communicate among themselves in order to exchange necessary data.
  - At the very beginning, before starting the main loop of iterations, each slave exchanges partial interface node values of local RHS vectors with all its neighbouring sub-domains. This exchange is performed using `MPI_Sendrecv`. Having all contributions from all the neighbours, a slave sums them up and gets global RHS values in shared nodes.
  - Each slave performs matrix-vector multiplications locally and then exchanges and adds contributions in shared nodes using `MPI_Sendrecv`, so that each slave has global product values in these nodes.
  - Each slave performs scalar vector products locally and then the master assembles and sums contributions from all the slaves using `MPI_Allreduce`, so that each slave has the global value of the calculated dot product of two vectors.
  - Each slave calculates linear combination of vectors, which is done locally without any communication. This operation is perfectly parallel.

It is clear that, in a FE implementation, only two kinds of communication between sub-domains are necessary. The first type of communication appears when matrix-vector multiplication is performed and consists of exchanging arrays between neighbouring sub-domains using `MPI_Sendrecv`. The necessity of this kind of communication can be explained through the principles of the domain decomposition technique on a simple example. Let me consider some domain Ω which is divided into two disjoint sub-domains Ω_1 and Ω_2 with the interface Γ_3, as shown in Fig. 3.4. After the mesh partitioning, which is based on the elements of the mesh,

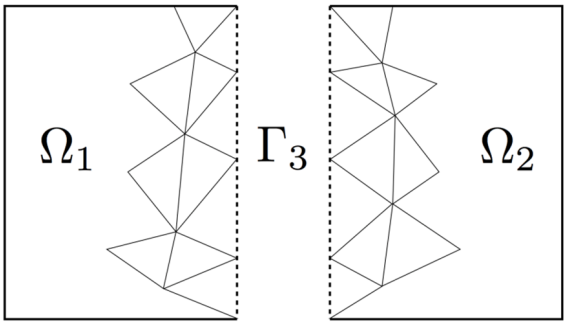


Figure 3.4: Decomposition of a domain into two sub-domains.

the nodes of the mesh can be renumbered according to the performed partition. In this way, the internal nodes of each sub-domain, as well as the interface nodes that are shared between the sub-domains, are grouped together within the global resultant linear system. In this simple case, as a result of renumbering, the internal nodes of sub-domains Ω_1 and Ω_2 have sub-indexes 1 and 2, respectively, while the interface nodes have sub-index 3. Thanks to this renumbering, the global system matrix, $\mathbf{A}$, can be rewritten as:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & 0 & \mathbf{A}_{13} \\ 0 & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix}, \tag{3.1}$$

where sub-matrices $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ represent all connections among the internal nods of sub-domains Ω_1 and Ω_2, respectively, sub-matrices $\mathbf{A}_{13}$,$\mathbf{A}_{23}$, $\mathbf{A}_{31}$, $\mathbf{A}_{32}$ describe how the internal nodes are connected to the interface nodes and sub-matrix $\mathbf{A}_{33}$ represents interactions between the nodes on the interface. Considering that in the FE context the entries of $\mathbf{A}_{33}$ reflect the values of the test functions of all the elements that share the nodes on the interface, this sub-matrix can be split into $\mathbf{A}_{33}^{(1)}$ and $\mathbf{A}_{33}^{(2)}$, which are contributions from sub-domains Ω_1 and Ω_2, respectively, so that:

$$\mathbf{A}_{33} = \mathbf{A}_{33}^{(1)} + \mathbf{A}_{33}^{(2)}. \tag{3.2}$$

Now, if each sub-mesh is given to a different process, Ω_1 to a process 1 and Ω_2 to a process 2, matrix assembly will be carried out in parallel. As a result, each process will create the corresponding part of the global resultant system matrix. These parts are, respectively:

$$\mathbf{A}^{(1)} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{13} \\ \mathbf{A}_{31} & \mathbf{A}_{33}^{(1)} \end{bmatrix} \qquad \mathbf{A}^{(2)} = \begin{bmatrix} \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{32} & \mathbf{A}_{33}^{(2)} \end{bmatrix}. \tag{3.3}$$

In this way, the global matrix is assembled locally in the sub-domains. It is clear that this part of the code is perfectly parallel and that, consequently, there is not any communication. Now, let me examine what happens when an iterative algebraic solver is executed. Every iterative solver performs matrix-vector multiplication, which is one of the basic operations of any algebraic iteration. When using the renumbered global system matrix, a generic matrix-vector product, $\mathbf{y} = \mathbf{A}\mathbf{x}$, is obtained as follows:

$$\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11} & 0 & \mathbf{A}_{13} \\ 0 & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix}, \tag{3.4}$$

$$\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{13}\mathbf{x}_3 \\ \mathbf{A}_{22}\mathbf{x}_2 + \mathbf{A}_{23}\mathbf{x}_3 \\ \mathbf{A}_{31}\mathbf{x}_1 + \mathbf{A}_{32}\mathbf{x}_2 + \mathbf{A}_{33}\mathbf{x}_3 \end{bmatrix}. \tag{3.5}$$

On the other hand, when an iterative method is executed in parallel, each process uses its local system matrix to perform matrix-vector multiplication, $\mathbf{y}^{(1)} = \mathbf{A}^{(1)}\mathbf{x}^{(1)}$ and $\mathbf{y}^{(2)} = \mathbf{A}^{(2)}\mathbf{x}^{(2)}$:

$$\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_3^{(1)} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{13}\mathbf{x}_3 \\ \mathbf{A}_{31}\mathbf{x}_1 + \mathbf{A}_{33}^{(1)}\mathbf{x}_3 \end{bmatrix} \qquad \begin{bmatrix} \mathbf{y}_2 \\ \mathbf{y}_3^{(2)} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{22}\mathbf{x}_2 + \mathbf{A}_{23}\mathbf{x}_3 \\ \mathbf{A}_{32}\mathbf{x}_2 + \mathbf{A}_{33}^{(2)}\mathbf{x}_3 \end{bmatrix}. \tag{3.6}$$

The interface nodes appear in both equations because they are repeated in each sub-domain, which is a consequence of the fact that mesh partitioning is based on the elements of the mesh. The idea of the domain decomposition is to recover the global result obtained in (3.5) each time the matrix-vector operation is performed. From (3.5) and (3.6), as well as $\mathbf{A}_{33} = \mathbf{A}_{33}^{(1)} + \mathbf{A}_{33}^{(2)}$, follows that:

$$\mathbf{y}_3 = \mathbf{y}_3^{(1)} + \mathbf{y}_3^{(2)}. \tag{3.7}$$

Hence, in order to have the same result in the interface nodes in Ω_1 as in the global case (3.5), contributions of Ω_2 in these nodes have to be added to $\mathbf{y}_3^{(1)}$. In the distributed memory context, this operation can be carried out by sending a massage that contains $\mathbf{y}_3^{(2)}$ from Ω_2 to Ω_1. The same thing stands for Ω_2. Therefore, the exchange of local shared node values can be carried out using `MPI_Sendrecv` between Ω_1 and Ω_2:

$$\Omega_2 \xrightarrow{\mathbf{y}_3^{(2)}} \Omega_1 \Longrightarrow \Omega_1 : \mathbf{y}_3^{(1)} \Leftarrow \mathbf{y}_3^{(1)} + \mathbf{y}_3^{(2)} \tag{3.8}$$

$$\Omega_2 \xleftarrow{\mathbf{y}_3^{(1)}} \Omega_1 \Longrightarrow \Omega_2 : \mathbf{y}_3^{(2)} \Leftarrow \mathbf{y}_3^{(2)} + \mathbf{y}_3^{(1)} \tag{3.9}$$

Taking all the above-mentioned into account, communication itself is performed in the following way. If two sub-domains share n nodes (interface nodes), they have to exchange arrays of size n. Namely, each of them has to send its local values in these nodes to the other one. The other one, after receiving the local values of its neighbour, adds these values to its own local values in shared nodes. In order to perform this communication, permutation arrays have to be used. These arrays find interface nodes in a vector, thanks to which the corresponding vector entries (values in shared nodes) are extracted in order to be sent. The same arrays are used when received values have to be added.

The second type of communication is global and of reduce type. This means that it is performed among all the slaves with `MPI_Reduce`. It is used to compute the scalar product of two vectors, which is one of the basic operations in iterative solvers. When calculating a scalar product, as well as the 2-norm (Euclidean norm) of a vector, each slave computes its portion of the final result, i.e. of the final sum. Every slave performs calculations for its internal nodes, as well as for its own interface nodes. After all the slaves have calculated their part of the

sum, communication is carried out with `MPI_AllReduce`, where the operation to be performed is `MPI_SUM`.

The second kind of communication is also used when managing a coarse system in algebraic multigrid (AMG) preconditioning. In the parallel context, when a distributed-memory machine is used, several techniques to deal with a coarse system exist (see e.g. Ramamurti & Löhner, 1996). In this work, there is only one, global, coarse matrix, although each slave contributes only to some of its entries. The global coarse matrix is obtained by simply carrying out `MPI_AllReduce` with `MPI_SUM` operation. This means that all the slaves have the complete coarse matrix and they all perform its factorisation. Then, when solving the coarse algebraic system, an additional `MPI_AllReduce` is required to assemble the complete RHS of the coarse system.

Speaking of inter-slave communication, a very important aspect that should be considered is the local node numbering of sub-meshes. In order to perform efficient data exchange, local nodes are divided into three categories, as illustrated in Fig. 3.5:

- Internal nodes – these nodes are not shared with another sub-domain. They are numbered using a METIS function that computes fill-reducing orderings of sparse matrices.
- Interface nodes – these nodes are shared with other sub-domains. Local values in these nodes must be exchanged among all the neighbouring sub-domains and summed up when computing the RHS vector or the matrix-vector product. Interface nodes are repeated over all the sub-domains that share them. Therefore, each sub-domain divides its interface nodes in Own and Others':
  - Own interface nodes are those for which the sub-domain itself is responsible. This is useful, for example, for computing scalar products, so that products in interface nodes are calculated only once before performing addition of contributions from all the sub-domains with `MPI_AllReduce`.
  - Others' interface nodes are those for which the neighbouring sub-domains are responsible.

The results of scalability tests with synchronous MPI communication are presented in Chapter 4, Subsection 4.4.1.

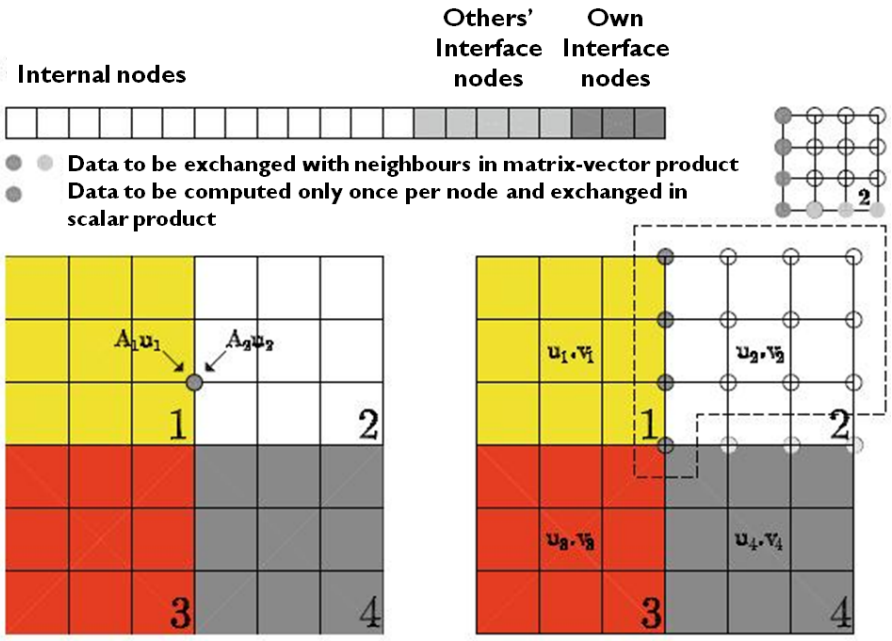


Figure 3.5: Local node numbering. From light to dark: internal nodes, interface nodes assigned to other sub-domains and own interface nodes.

3.2.2 Asynchronous Communication

When performing matrix-vector multiplication, slaves need to communicate in order to exchange values of the product vector in the nodes that they share. When synchronous MPI communication is used, matrix-vector multiplication is performed in the following way:

- First, each slave performs the complete matrix-vector multiplication by executing a loop that iterates through all the nodes (internal and interface nodes) that belong to the corresponding assigned sub-domain. As a result, each slave obtains the whole product vector with calculated values in both the interior and interface (shared) nodes. However, the values in the shared nodes are only partial, because of which it is necessary to add to them the results obtained by the corresponding neighbours.

- After obtaining the product vector, each slave exchanges the calculated values that belong to the shared nodes with the corresponding neighbours. For this, synchronous (blocking) `MPI_Sendrecv` function is used.

- Finally, after receiving and summing up all contributions from the neighbours, each slave has the complete resultant vector that can be used for further computations.

When asynchronous MPI communication is employed, matrix-vector multiplication is performed in the following way:

- First, each slave performs only one part of matrix-vector multiplication by executing a loop that iterates only through the interface (shared) nodes. As a result, each slave obtains only a part of the product vector that corresponds to the shared nodes. These calculated values are only partial and need to be exchanged among the corresponding neighbours and summed up.
- After obtaining one part of the resultant vector that has to be exchanged, each slave initiates asynchronous communication with the corresponding neighbours employing non-blocking `MPI_Isend` and `MPI_Irecv` functions.
- While communication proceeds, each slave continues to perform the rest of matrix-vector multiplication and obtains the part of the product vector that corresponds to its internal nodes, whose values do not need to be exchanged. To do this, each slave performs a loop that iterates through the internal nodes.
- After this, it is necessary that each slave waits for all the requested contributions to arrive from the corresponding neighbours – at this point `MPI_WAITALL` is used for synchronisation – before summing them up with the values calculated by the slave itself at the beginning of the procedure.
- Finally, all the slaves have complete resultant vectors that can be used for further computations.

The results of scalability tests with asynchronous MPI communication are presented in Chapter 4, Subsection 4.4.1.

3.3 Hybrid Parallelisation using OpenMP

In addition to MPI, which is used for communication among processes, I have employed OpenMP for parallelisation within each process. In this way, a hybrid parallel scheme has been created. In order to perform OpenMP parallelisation, I have used compiler-directive-based OpenMP programming model, which means that OpenMP – multi-threaded, shared-memory – parallelism has been specified by the use of compiler directives which are embedded in the source code.

As explained in Chapter 2, Section 2.5, when using AMG preconditioning, it is necessary to create a coarse system. When slaves execute an iterative solver preconditioned with AMG, they have to compute the coarse matrix of the system as well as to factorise it using LU factorisation

before the main loop of iterations. In the previous section, Subsection 3.2.1, it is told that, in this work, there is only one, global, coarse matrix that exists in each computational unit. This means that each slave has the whole coarse matrix and performs its complete LU factorisation. However, LU factorisation is rather time-consuming part of the code, depending on the size of the coarse matrix. The bigger the matrix is, the more time is needed for its factorisation. For example, LU factorisation of a coarse matrix whose size is 40,000×40,000, which means that it is created from 10,000 groups, takes 75% of the total execution time. Also, for other normally used sizes of the coarse matrix, this part of the code is dominant in terms of execution time. Therefore, in order to reduce its time of execution, I have employed OpenMP for its parallelisation within each computational unit.

OpenMP parallelisation of LU factorisation of a complex sparse matrix stored in the compressed sparse row (CSR) format has been performed in two steps:

- The first step is symbolical factorisation, which computes the structure of L and U matrices. This routine is a single-thread routine.
- The second step is numerical factorisation, which computes the entries of L and U matrices. This routine is a multiple-thread routine.
  - In the sequential version, the main factorisation loop iterates through the rows of the matrices. However, in the parallel, multi-threaded, version, the outer loop iterates through the packets of rows – which are created using the elimination tree of the factorisation. Iterations of this outer loop are then distributed dynamically among the threads – one iteration (packet of rows) at a time.
  - In order to determine the execution order of iterations of the factorisation loop in the multi-threaded version of numerical factorisation, the following steps are performed:
    * First, the elimination tree of the factorisation (Gilbert & Liu, 1993) is computed. The nodes of this tree represent rows of the matrix, while their connections show row dependencies during the factorisation.
    * Then, blocks (packets) of nodes (rows) are created. The number of packets is equal to the number of threads. The algorithm tries to make these blocks as big as possible, but at the same time equally balanced. The rest of the nodes, which are not packed in the created blocks, are packed in packets of only one node. In this way, each node belongs to some packet.
    * After these packets of nodes (rows) have been created, they can be distributed among the threads so that each thread executes iterations connected to the rows

(nodes) that belong to the assigned packets. The algorithm assigns dynamically one packet of nodes at a time by distributing iterations of the outer loop among threads using:

```
#pragma omp for schedule(dynamic,1)
for (bb= 0;  bb< number_of_packages;  bb++)
{
 numerical factorisation
}
```

The results of scalability tests when the hybrid MPI–OpenMP scheme is employed are presented in Chapter 4, Subsection 4.4.2.

Chapter 4

Evaluation and Discussion

4.1 Accuracy Tests

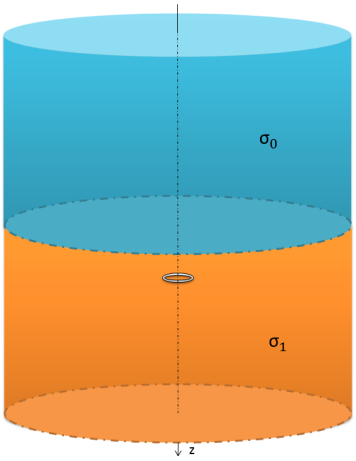
I have performed several tests in order to check the accuracy of the presented finite-element method. All models that I have used in the tests demonstrated here are synthetic, due to the fact that I am not allowed to present real data since they are strictly confidential property of oil companies.

4.1.1 Two-Layer Model

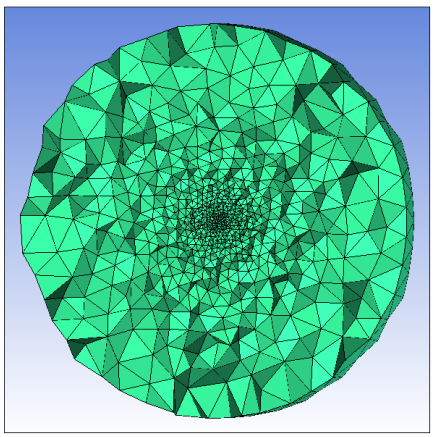
The first model, given in Fig. 4.1a, is the one-dimensional (1-D) model of a two-layer geoelectrical structure proposed by Badea *et al.* (2001). The source is a horizontal finite current loop, whose radius is 0.01 m, placed at 1.5 m below the interface that separates two conductive materials. The source carries an electric current of 10^{10} A that oscillates at 2.5 MHz. I remark that this case is a borehole problem and thus the frequency is considerably large compared to frequencies normally used in CSEM. The skin depth in the lower, less conductive, half-space, which contains the source, is $z_{s0} = \sqrt{\frac{2}{\sigma_0 \mu_0 \omega}}$ [m]. Expressions for the primary potentials associated to this kind of source located in a homogeneous medium can be found in Badea *et al.* (2001).

The problem domain is a cylinder whose radius is 9 m and length is 20.35 m. The mesh that has been generated for this model has a strong refinement near the source and on the z-axis, along which the electric field varies rapidly. A slice of the mesh in the X–Y plane at $z = 1.5$ m is shown in Fig. 4.1b. This mesh has 543,319 elements and 93,406 nodes, which means that the system to be solved has 373,624 unknowns.

The numerical results obtained at the symmetry axis of the model have been compared with the analytical solution to the problem (Ward & Hohmann, 1988).



(a) Model of a two-layer geo-electrical structure with a horizontal finite current loop as the source.



(b) Slice of the mesh in the X-Y plane.

Figure 4.1: Two-layer model and a slice of the mesh generated for it

Small Conductivity Contrast

For the first test, the conductivities of the lower and the upper half-space are $\sigma_0 = 0.1$ S/m and $\sigma_1 = 1$ S/m, respectively, which means that the conductivity contrast in this case is quite modest: $\sigma_1/\sigma_0 = 10$.

The vertical component of the magnetic field numerically calculated along the z-axis, H_z, has been compared to the corresponding analytical solution (Ward & Hohmann, 1988) in Fig. 4.2. It is clear that the numerical solution is in agreement with the analytical one to a great extent.

Big Conductivity Contrast

Secondly, I have performed the same test, using the same mesh, for a much bigger conductivity contrast: $\sigma_1/\sigma_0 = 10^5$.

Fig. 4.3 illustrates that obtained numerical results very precisely match the analytical ones, which proves that presented approach is able to deal with strong conductivity contrasts that often appear in realistic models.

Mesh With Local Refinements vs. Uniform Mesh

Furthermore, I have compared the results obtained using a mesh that has two levels of refinement with the results acquired when an uniform mesh without refinements has been employed. The refined mesh has one level of refinement in the ellipsoidal region defined by $0 \leq r \leq 2.5$ m and

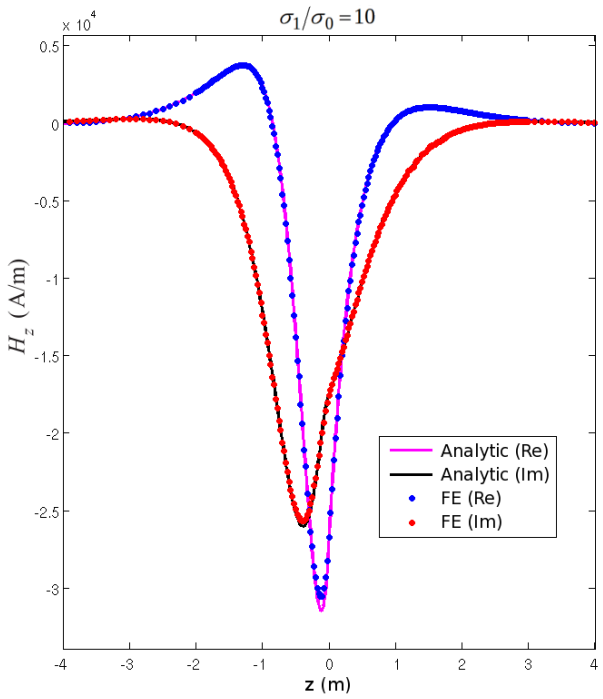


Figure 4.2: Comparison of the FE solution with the analytical one. The solid lines represent real (light line) and imaginary (dark line) parts of the analytical solution. The numerical solution is marked with circles.

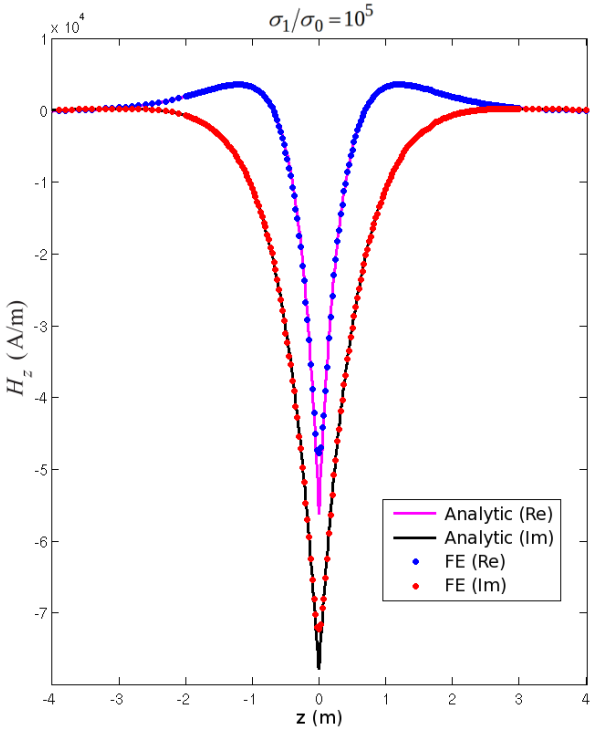


Figure 4.3: The same as Figure 4.2, but for the conductivity contrast of 10^5.

$-4.5 \leq z \leq 4.5$ m, while the other refinement level is in the region described by $0 \leq r \leq 1.0$ m and $-3.0 \leq z \leq 3.0$ m. This mesh has 287,152 nodes and 1,677,390 elements whose sizes vary from 0.5 m, near the edges of the domain, to 0.1 m, near the centre. The uniform mesh has 489,239 nodes and 2,935,432 elements of size equal to 0.25 m. These meshes have been created in such a way that the average size of the elements is almost the same in both cases.

The solutions for these two meshes are marked in Fig. 4.4 with circles and diamonds, respectively, while the complex analytical solution is represented by the solid (real part) and dashed (imaginary part) curves. It can be clearly seen that the results for the mesh with local refinements are much more precise near the source ($z = 1.5$ m) and at the peaks near the interface ($z = 0$ m) compared with the results for the uniform tetrahedral mesh with the smaller element size in the whole domain, but without refinements in the centre. The least-square distance between the analytical solution and the solution for the refined mesh is 5.2 times smaller than the distance for the uniform mesh. For the uniform mesh, convergence is achieved in 70 itera-

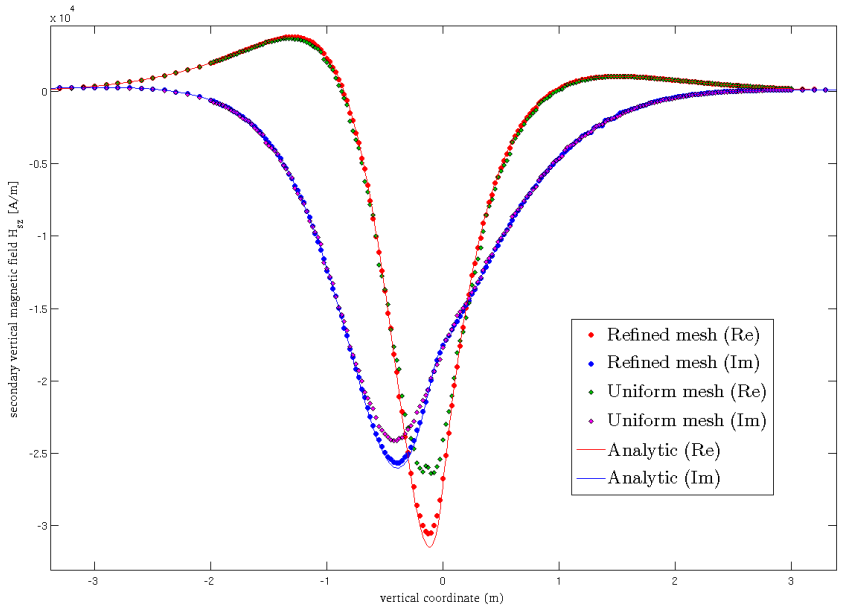


Figure 4.4: Comparison of the solutions for the mesh with two levels of refinement (circles) and for the uniform mesh (diamonds).

tions and total (sequential) CPU time of execution is 820 s. Using the mesh with two levels of refinement, the program converges in 160 iterations and total execution time is 399 s.

This experiment shows the influence of a mesh on the quality of a solution, numerical features of a solver as well as efficiency of the whole program. The conclusion is that a non-uniform mesh with local refinements has less elements, produces highly precise results, but causes a slower convergence, while an uniform mesh without refinements has more elements, generates less

accurate results, but produces a faster convergence. However, despite the slower convergence, non-uniform meshes normally provide faster code executions since they have less elements and consequently each iteration is less costly.

4.1.2 Flat Seabed Model

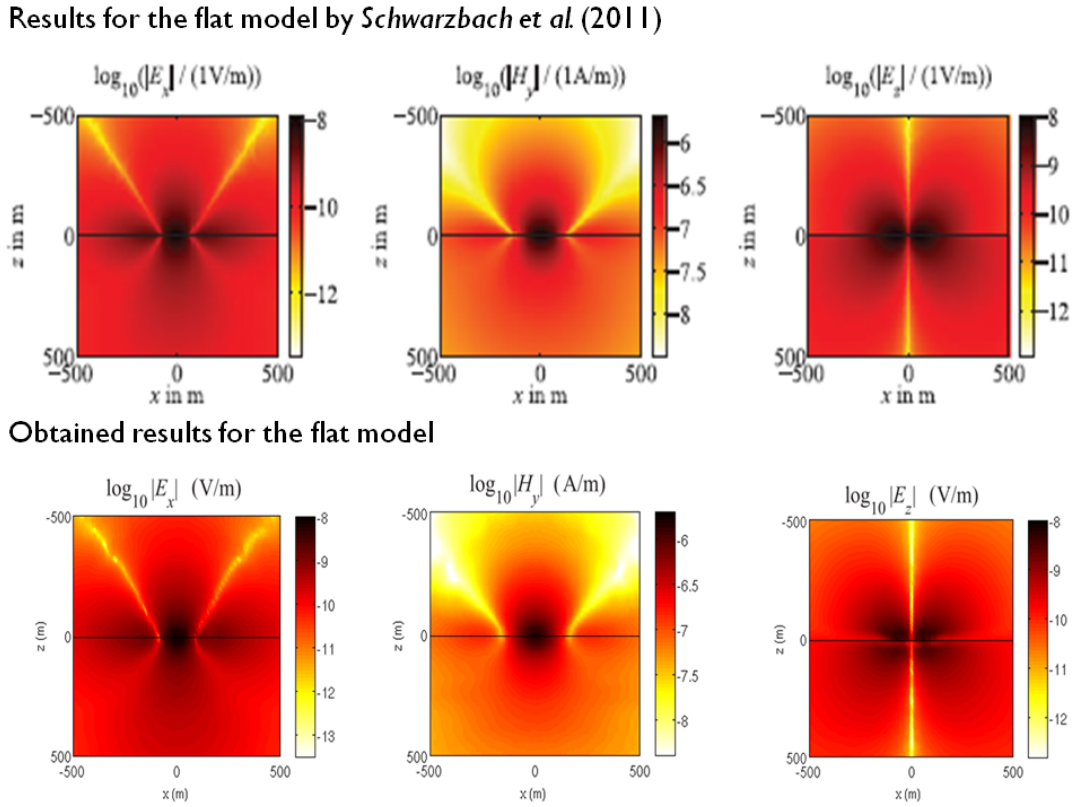
The next numerical example is the flat seabed model described by Schwarzbach *et al.* (2011). Similarly to the two-layer, this model also has a simple geometry. It consists of two half-spaces, which represent seawater ($\sigma_0 = 3,3$ S/m) and sediments ($\sigma_1 = 1,0$ S/m), separated by a planar interface. Unlike the previous case, the source here is an x-directed horizontal electric dipole that radiates with frequency of 1 Hz and is placed 100 m above the seabed. The dipole strength (the product of dipole peak current and dipole length) is 1 Am. For a finite dipole whose length is small compared to the source-receiver separation, this parameter is proportional to the amplitude of the field detected by the receivers. The computational domain, $\omega = \{-2 \text{ km} \leq x, y \leq 2 \text{ km}; -1.5 \text{ km} \leq z \leq 2.5 \text{ km}\}$, has been chosen according to Schwarzbach *et al.* (2011) in order to compare my results with those reported in the paper. The background conductivity model has been considered homogeneous with the conductivity of the seawater, $\sigma_0 = 3.3$ S/m. This is a common setup for marine CSEM.

The mesh created for this model has 3,121,712 elements and 533,639 nodes, and hence the system to be solved has 2,134,556 degrees of freedom. The average size of the elements ranges from 6 m, near the source, to 100 m, at the boundaries of the domain.

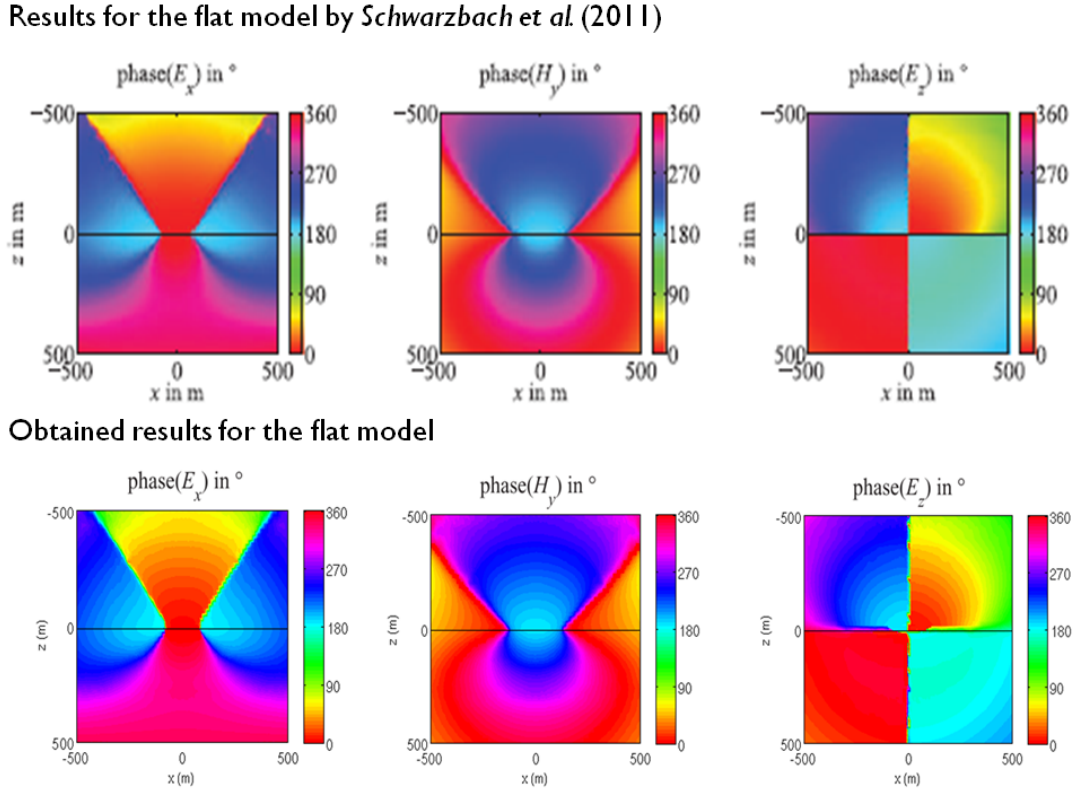
The numerically obtained absolute values and phases of non-vanishing secondary electric- and magnetic-field components in the X–Z plane at $y = 0$ are shown in Fig. 4.5. These results are remarkably similar to those presented in Fig. 5 by Schwarzbach *et al.* (2011) (these results are also given in Fig. 4.5). The other three components, H_x, E_y and H_z, should vanish along an in-line profile through the centre of the model because of the symmetry. Obtained results for these components contain only numerical noise, which is 3–4 orders of magnitude lower than the values given in Fig. 4.5.

4.1.3 Canonical Disc Model

In this subsection, simulations for the canonical disc model proposed by Constable & Weiss (2006) are presented. This model was recently studied by Schwarzbach *et al.* (2011), who compared the results of their FE simulation with those of the Weiss's finite-volume code FDM3D. Also, they added an air half-space in the model in order to analyse the air-wave effect. The canonical disc model, shown in Fig. 4.6a, consists of two half-spaces, which represent 1.5 km deep seawater ($\sigma_0 = 3.3$ S/m) and sediments ($\sigma_1 = 1.0$ S/m), and a disc which is a simplified



(a) Absolute values of non-vanishing secondary field components in the X-Z plane.



(b) Phases of non-vanishing secondary field components in the X-Z plane.

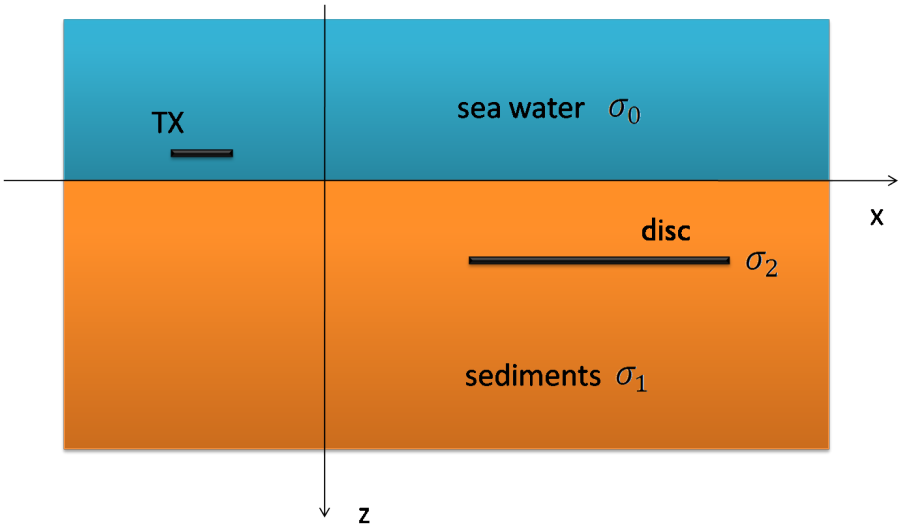
Figure 4.5: Obtained solution for the flat seabed model compared to the solution published by Schwarzbach *et al.* (2011).

model of a hydrocarbon reservoir ($\sigma_2 = 0.01$ S/m). The disc, whose radius is 2 km and hight is 100 m, is located at 1 km beneath the interface. The transmitter is a horizontal electric dipole oriented in the x-direction. The dipole is placed at the spot with coordinates (-3000, 0, -100) and operates at the frequency of 1 Hz. The model with the air half-space, shown in Fig. 4.6b, has the same characteristics, except that the thickness of the water layer is 1 km and the computational domain is much larger. As noted by Schwarzbach *et al.* (2011), the results are largely influenced by the choice of the background conductivity model for the primary field. Therefore, I also have chosen a homogeneous model in order to be able to compare the results.

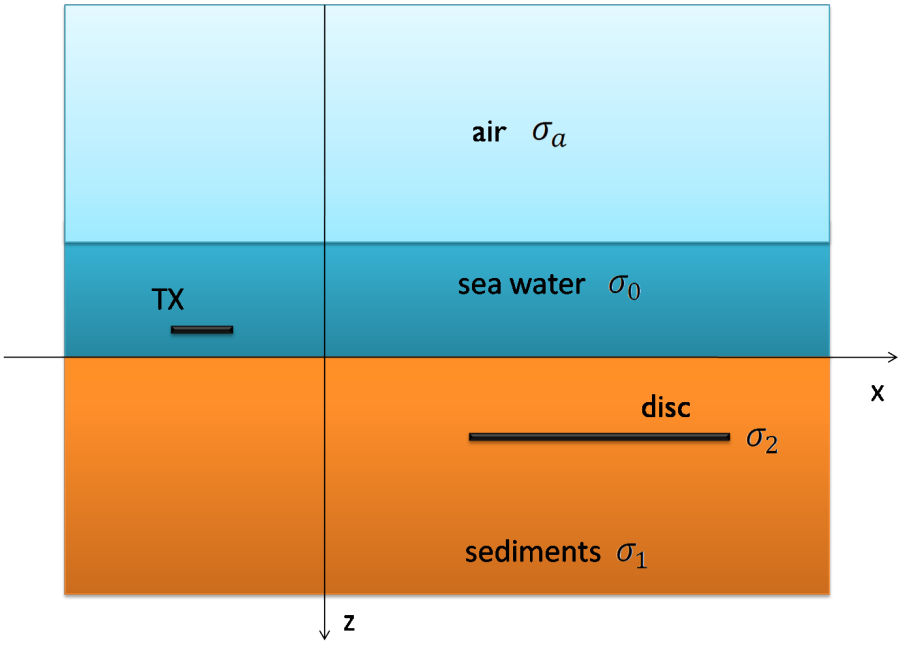
The model responses obtained using the presented FE method are presented in Fig. 4.7a. The EM-field values have been calculated along an in-line profile through the centre of the model for four different scenarios. Namely, for both versions of the model – with and without the air half-space – two simulations have been performed – with the hydrocarbon reservoir and without it ($\sigma_2 = \sigma_1$). Due to the fact that the EM field decays rapidly in a conductive medium, the secondary field usually is much smaller than the primary one. In many cases, this leads to the situation in which the difference between the results of simulations with and without a hydrocarbon reservoir is quite small and the target field is partially hidden in the background field. Fig. 4.7a, just like Fig. 9 published by Schwarzbach *et al.* (2011) (here given in Fig. 4.7b), shows that these fields are indistinguishable for transmitter-receiver distances smaller than 2.5 km. Further comparison with the results obtained by Schwarzbach *et al.* (2011) shows that both numerical solutions are in agreement to a great extent. Namely, in both cases, the fields decay slower if the resistive disc is included in the model. Also, all three non-vanishing field components are the same for offsets smaller than 3 km. Moreover, the presence of the air half-space has a small effect on the vertical electrical component. However its influence is significant for the horizontal field components at offsets larger than 4.5 km. Both meshes used for simulations with and without the air half-space have approximately 3 million elements and 0.5 million nodes (2 million degrees of freedom).

4.1.4 Complex Real-Life Synthetic Model

In order to test possibilities of the presented FE approach to deal with arbitrarily complex geological structures, a large realistic synthetic test case that includes seabed bathymetry, shown in Fig. 4.8, has been created. If not taken into account, bathymetry effects can produce large anomalies on the measured fields. Therefore, it is extremely important to have an accurate seabed representation in this case. The dimensions of the model are $15\times12\times6.2$ km and the water depth varies from 1,050 to 1,200 m. The subsurface has 5 different anisotropic conductivity structures with σ_h ranging from 0.1 to 1.5 S/m and σ_v varying from 0.12 to 1.0 S/m. The

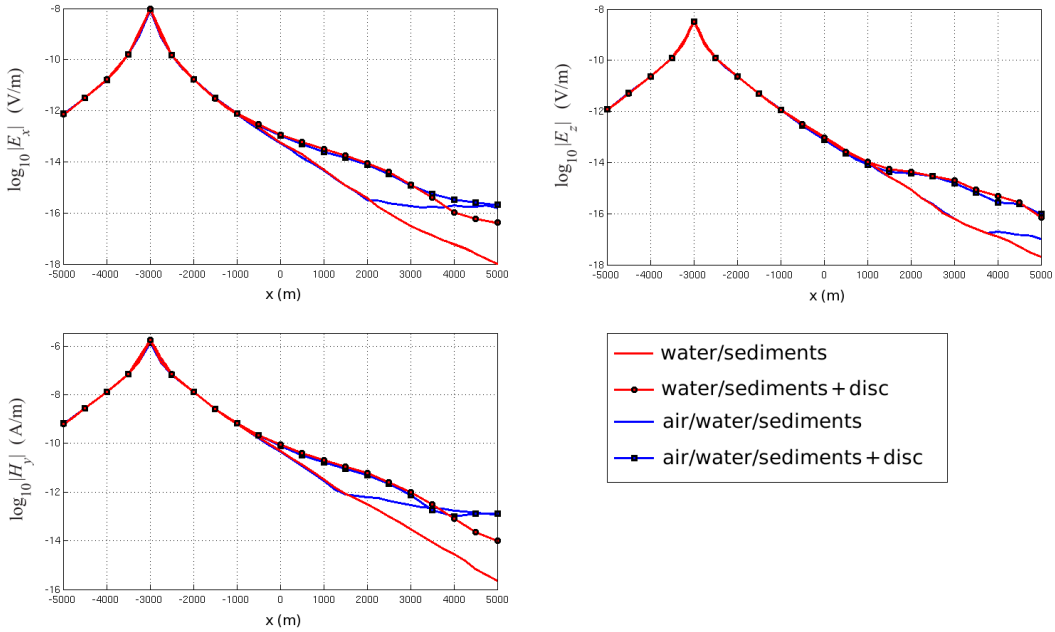


(a) Canonical disk model without the air layer.

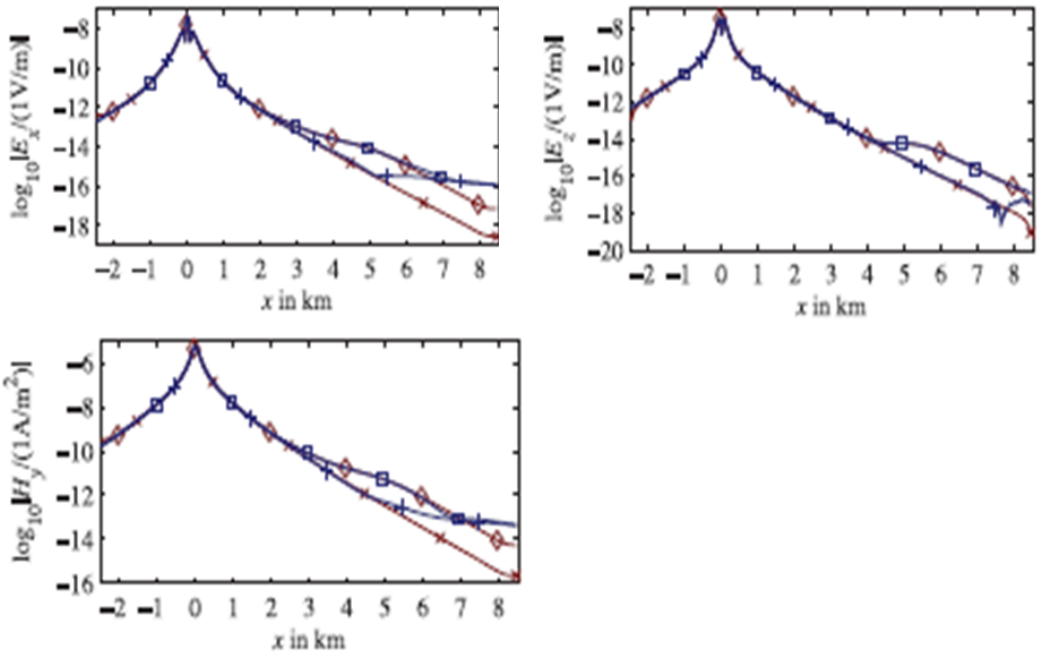


(b) Canonical disk model with the air layer.

Figure 4.6: Canonical disk model consisting of two half-spaces and a disk, which represents a hydrocarbon reservoir, with a x-oriented horizontal electric dipole as the source, without and with the air layer.

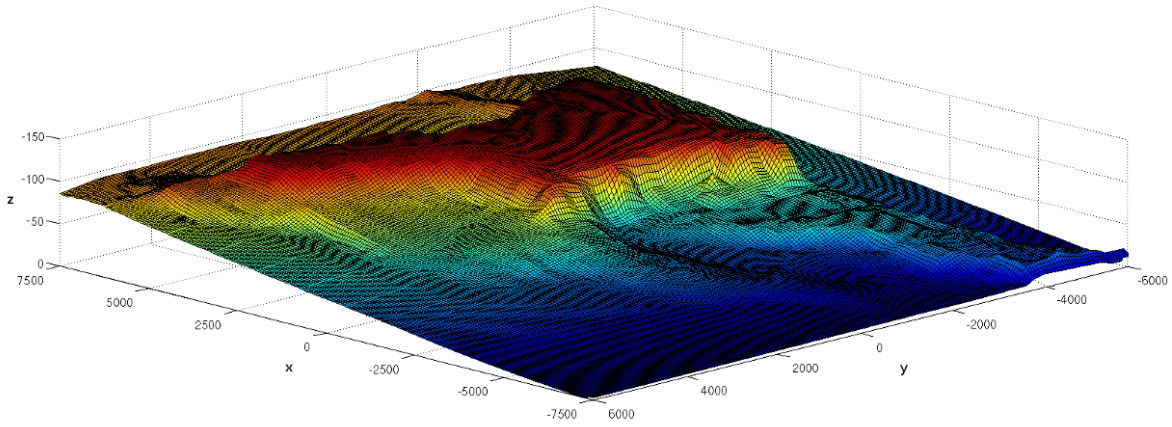


(a) Obtained non-vanishing secondary field components.

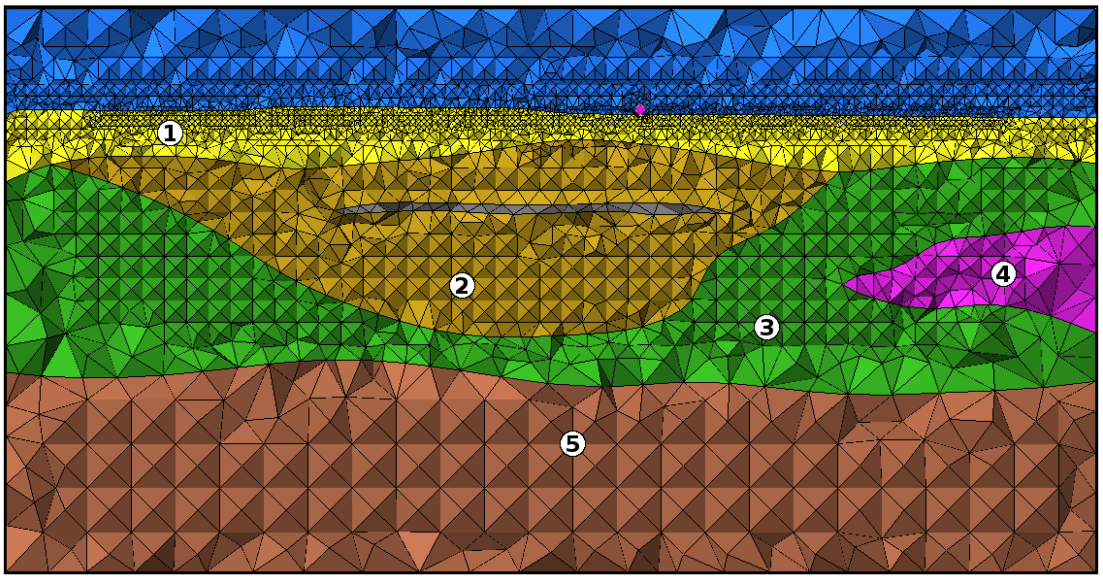


(b) Non-vanishing secondary field components published by Schwarzbach *et al.* (2011).

Figure 4.7: Obtained solution for the canonical disc model compared to the solution published by Schwarzbach *et al.* (2011).



(a) Seabed bathymetry.



(b) A X-Z slice.

Figure 4.8: Complex real-life synthetic model.

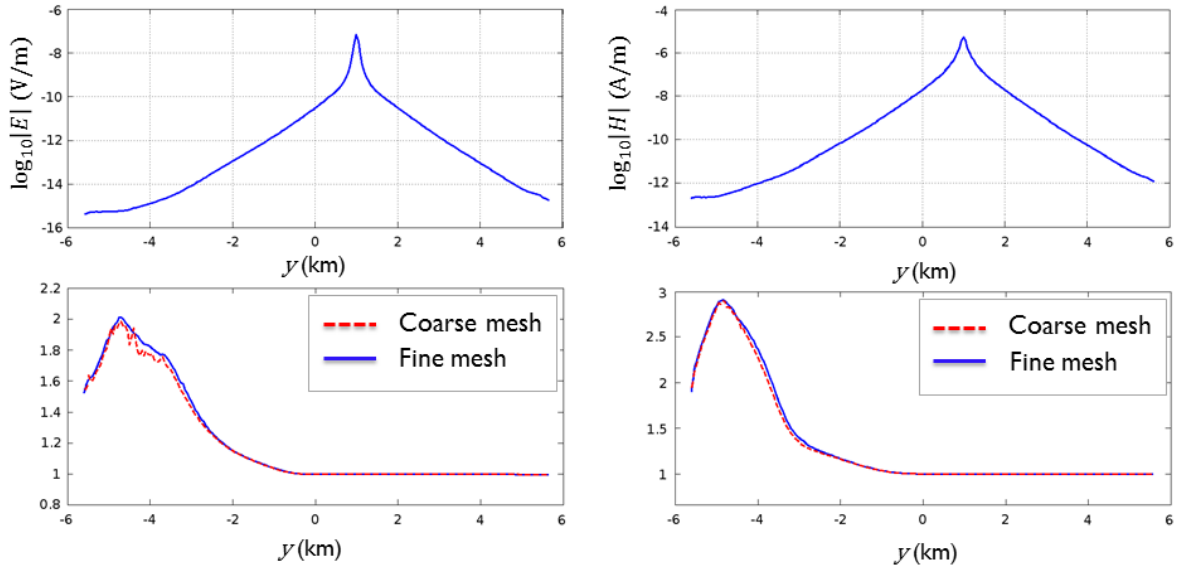


Figure 4.9: Secondary fields for the complex synthetic model: absolute values (top) and values normalized by the background results (without the reservoir)(bottom).

hydrocarbon reservoir is located at 1000 m below the seabed and has the conductivity $\sigma_h = \sigma_v = 0.01$ S/m. The transmitter is a horizontal electric dipole operating at the frequency of 1 Hz. In order to accurately represent the complex geology and seabed bathymetry, A mesh that has 512,651 nodes and 2,996,017 elements, whose sizes vary from 6 to 400 m, has been created. Also, in order to analyse the importance of the quality of a mesh, another, smaller, mesh that has 1.3 million elements ranging from 10 to 400 m has been made.

The model responses along an in-line profile are presented in Fig. 4.9. The upper row shows the absolute values of the secondary fields, while the bottom row demonstrates secondary electric and magnetic field magnitudes normalised by the corresponding results of the simulation without the reservoir. It is clear that the results for the two meshes are of a similar order of magnitude. However, the results obtained with the finer mesh do not include numerical oscillations that appear in the results achieved with the coarse mesh.

4.2 Convergence of the Solvers

In this section, I study the performance of the implemented solvers – the right-preconditioned BiCGStab, QMR and GMRES – with the Jacobi (diagonal) preconditioner in order to compare their behaviour for the EM problem under consideration and to determine which one is the best choice in this case. This is due to the fact that different linear solvers can compete in terms of

convergence rate, computational cost of each iteration and memory requirements. To perform the convergence tests, I have used the meshes created for the models described in Section 4.1. In all tests, the convergence criterion is a reduction of the relative residual norm to a value in the order of 10^{-8}. It is important to emphasise that QMR minimises the norm of the quasi-residual (Simoncini & Szyld, 2007) instead of the residual norm that is directly generated in case of GMRES and BiCGStab. This means that the convergence criterion is applied to this quasi-residual and not to the real one, which is important and relevant for fair comparisons to the other methods. Therefore, real QMR residual norms have to be calculated in order to evaluate its true convergence. Also, the number of iterations has been limited by the maximum value of 1,000.

The convergence plot in Fig. 4.10 shows the residual norms generated by BiCGStab and QMR versus the iteration number for the mesh created for the model without the air-layer described in Subection 4.1.3 which has 2,993,420 elements and 512,060 nodes (2,048,240 degrees of freedom). Fig. 4.10 shows the norms of both real and quasi- residuals generated by QMR in order to compare its convergence to the one of BiCGStab. The convergence plot in Fig. 4.11

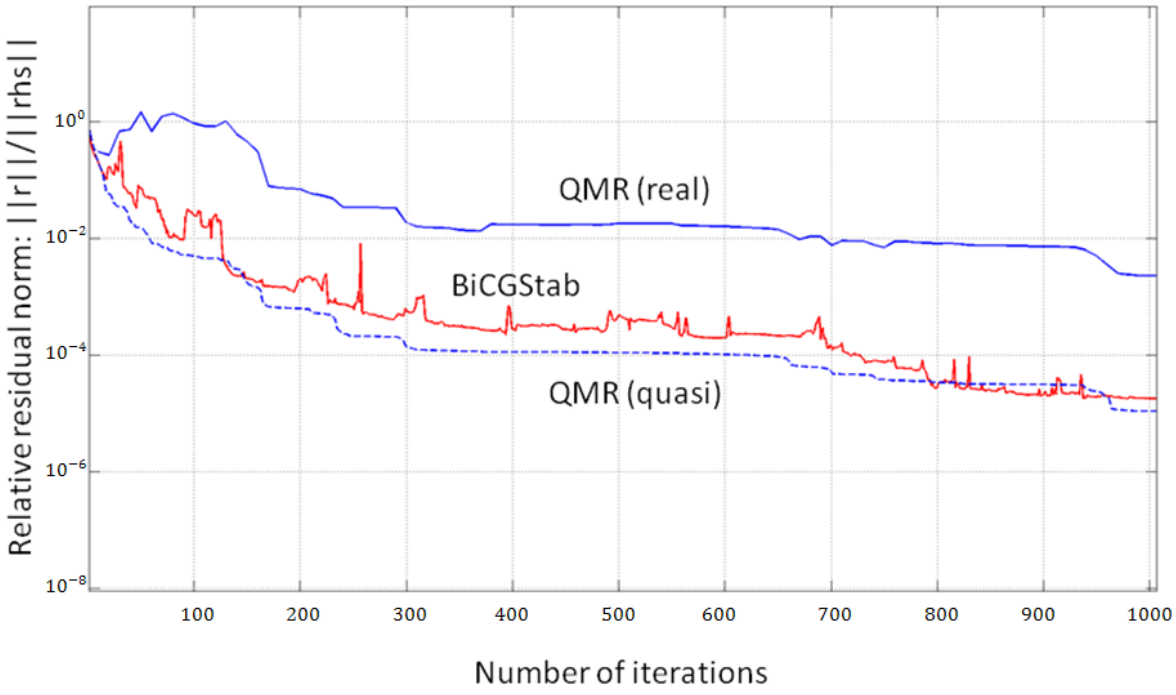


Figure 4.10: Convergence rate of BiCGStab and QMR solvers for the mesh of 2 million degrees of freedom created for the canonical disk model without the air.

shows the norms of the BiCGStab residuals and QMR real and quasi- residuals versus the

iteration number for the big mesh created for the model described in Subection 4.1.4 which has 2,996,017 elements and 512,651 nodes (2,050,604 degrees of freedom). We can see that in

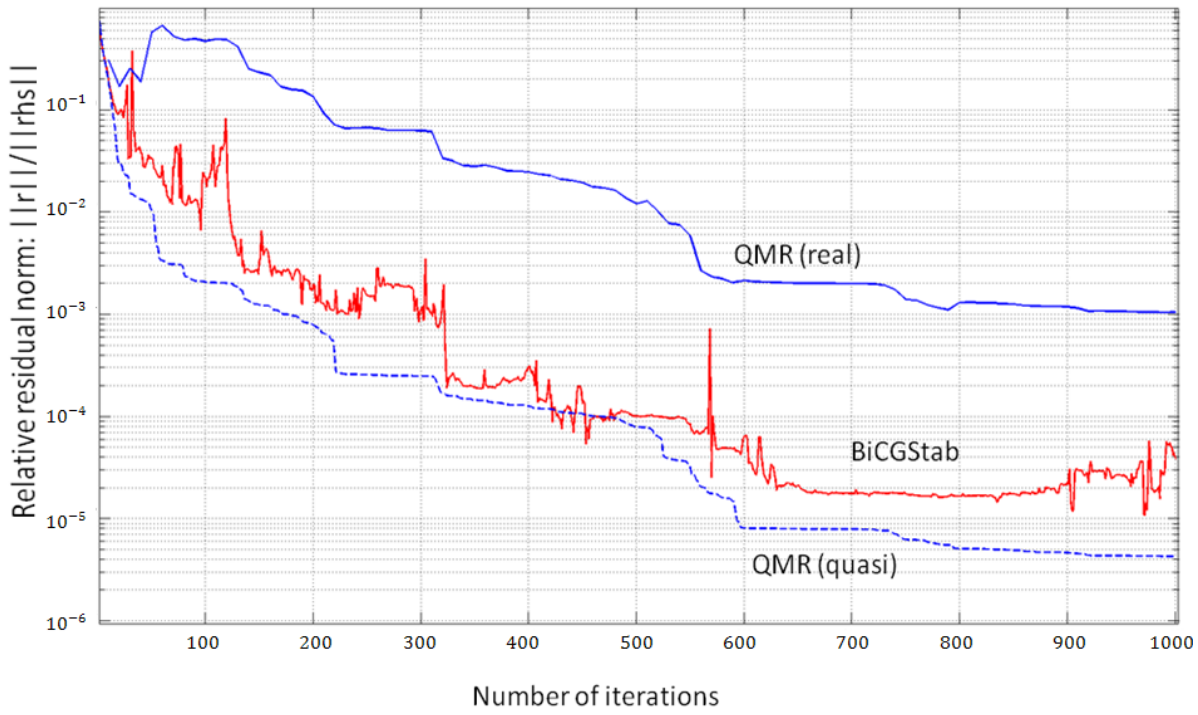


Figure 4.11: Convergence rate of BiCGStab and QMR solvers for the mesh of 2 million degrees of freedom created for the real-life synthetic model.

both cases BiCGStab has more dramatic oscillations, however, its overall convergence is faster and it can reach lower residual norms after the same number of iterations: in the first case, it reaches 10^{-4} vs. 10^{-2} of QMR, while in the second case, BiCGStab reaches 10^{-4} and QMR 10^{-3} after 1,000 iterations. Also, the convergence of both methods in both examples is quite poor since neither of the methods can reach the given precision of 10^{-8} in 1,000 iterations using only diagonal preconditioner. Namely, due to the big sizes of the domains, both meshes have large aspect ratios of the elements, which results in bad convergence rate. This is quite common situation in practice where problem domains are huge and in order not to have too many elements and to obtain an accurate solution unstructured meshes are refined in the vicinity of a source and receivers and aspect ratios of the elements is usually very big which causes slow convergence rate. It can be concluded that a more elaborate and more powerful preconditioning technique for complex Helmholtz-type equation systems instead of the simple diagonal one is essential for improving the convergence rate. More tests have confirmed the results presented here, as well as that QMR, quite popular solver in the community, can reach BiCGStab after

several hundreds or thousands of iterations, though at the beginning its convergence rate is much worse. Thus, for the problem under consideration, BiCGStab appears to be a better choice.

The residual norm of GMRES without restarts decreases monotonically and the convergence is smooth, however BiCGStab with its oscillations performs better or equal. Due to the large memory requirements of GMRES, there seems to be no reason to use it in real simulations.

4.2.1 Discussion

Since linear systems arising from the EM problem under consideration are huge, GMRES has proved to be highly impractical because of its large demands for memory. Also, the experiments, which I have carried out to test the performances of the implemented solvers for these systems, have shown that BiCGStab produces norms of the residuals that oscillate significantly, but are smaller in value than the ones of GMRES and QMR. Therefore, for this class of problems I suggest using BiCGStab in combination with a proper preconditioner.

4.3 AMG Evaluation

In order to evaluate the presented AMG preconditioning scheme, I have performed various tests for several Earth models described above. I have chosen models with different conductivity contrasts – from small ones, in the order of 10, to quite high contrasts, in the order of 10^5. It is important to test a preconditioning scheme for cases with high contrasts between electrical conductivities since solvers normally have convergence problems when dealing with such conductivity structures. Also, in different tests, employed frequencies have different values – from low frequencies ($\sim$1 Hz), which CSEM surveys usually use, to high frequencies ($\sim 10^6$ Hz). Furthermore, in some tests, the source is a current loop, while in others, it is an electric dipole.

In all the following experiments, I have used the right-preconditioned BiCGStab method to solve linear systems that are results of the FE discretisation. This iterative solver has been chosen because the earlier tests, which have been carried out in order to examine the behaviour of the implemented solvers, have shown that, for the EM problem under consideration, the norms of the residuals produced by BiCGStab oscillate quite a lot, but are smaller than the ones of GMRES and QMR. The convergence criterion for all BiCGStab iterations is a reduction of the relative residual norm to a value in the order of 10^{-10}. Also, the number of iterations has been limited by the maximum value of 3,000. All the execution times have been obtained by running the program in parallel on the **MareNostrumIII supercomputer** using 32 processes.

To inspect the benefits of the new preconditioning scheme, the performances of several preconditioning strategies have been compared:

1. Jacobi (diagonal) preconditioning
2. SSOR preconditioning with over-relaxation parameter equal to 0.1
3. AMG preconditioning with Jacobi smoother (AMG-J)
4. AMG preconditioning with SSOR smoother (AMG-SSOR)

The AMG preconditioner has been tested with different parameters:

- number of basic iterations at the beginning and at the end of the CGC procedure: 1, 3
- number of groups, i.e. size of the coarse system: 100, 500, 1000, 5,000, 10,000

4.3.1 Two-Layer Model

The first model is the 1-D model described in Subsection 4.1.1.

Small Conductivity Contrast

First, I analyse the case when the model has a quite small conductivity contrast, $\sigma_1/\sigma_0 = 10$, as given in Subsection 4.1.1. For this case, I present the results for both AMG-J and AMG-SSOR, and for all combinations of the AMG parameters given above. Table 4.1 shows the convergence, given in number of iterations, of the preconditioned BiCGStab solver, as well as the total execution time of the code, expressed in seconds, when employing AMG-J and AMG-SSOR preconditioning with different parameters.

Looking at these results, some conclusions can be drawn. First, it can be noticed that if the number of basic iterations at the beginning and at the end of the CGC procedure is increased, the convergence of the solver improves. Also, if more groups are created, the solver converges faster. Furthermore, it is clear that the choice of smoother affects the solver's convergence rate. It can be seen that, in this test, SSOR smoothing gives the smallest achieved number of iterations, 138, which is reached for the largest used number of basic iterations (3) and the largest used number of groups (5,000), while Jacobi smoothing leads to the better convergence on the average (392.5 vs. 712.6 iterations).

However, the presented results show that if the convergence is improved by increasing AMG parameters, it does not mean that the execution time of the code will be reduced. Namely, one AMG-preconditioned iteration is quite costly and each increment of any parameter makes it even more expensive. In addition, SSOR iterations are more computationally expensive than Jacobi

No. of SSOR iterations	3+3	3+3	3+3	3+3	1+1	1+1	1+1	1+1
No. of groups	100	500	1000	5000	100	500	1000	5000
Convergence (No. of iterations)	1309	499	291	138	1694	874	725	171
Total execution time (sec.)	77.44	32.83	22.81	46.57	79.18	30.01	32.54	47.35
No. of Jacobi iterations	3+3	3+3	3+3	3+3	1+1	1+1	1+1	1+1
No. of groups	100	500	1000	5000	100	500	1000	5000
Convergence (No. of iterations)	482	387	306	215	733	421	381	215
Total execution time (sec.)	26.34	23.31	21.53	61.64	20.96	14.37	16.99	55.81

Table 4.1: Results for the two-layer model with the small conductivity contrast.

iterations. Therefore, care has to be taken when choosing the AMG parameters in order to get the best possible performance. Sometimes, it is necessary to chose more expensive elements in order to make the solver converge to a desired precision. On the other hand, sometimes, it is necessary to select a cheaper version which provides the fastest solution, although it may not give the best convergence. Considering this model, for example, the shortest execution time of 14.37 s is obtained using 1+1 Jacobi iterations and 500 groups.

In Fig. 4.12, I compare the convergence of the solver without any preconditioner and with different preconditioning schemes, including the AMG-J preconditioner with 1+1 basic iterations and variant number of groups. The chart shows relative norms of the residuals generated by BiCGStab iterations, $\|\mathbf{r}\|/\|\mathbf{b}\|$ (where $\|\mathbf{b}\|$ is the Euclidean norm of the RHS vector), as a function of the number of iterations.

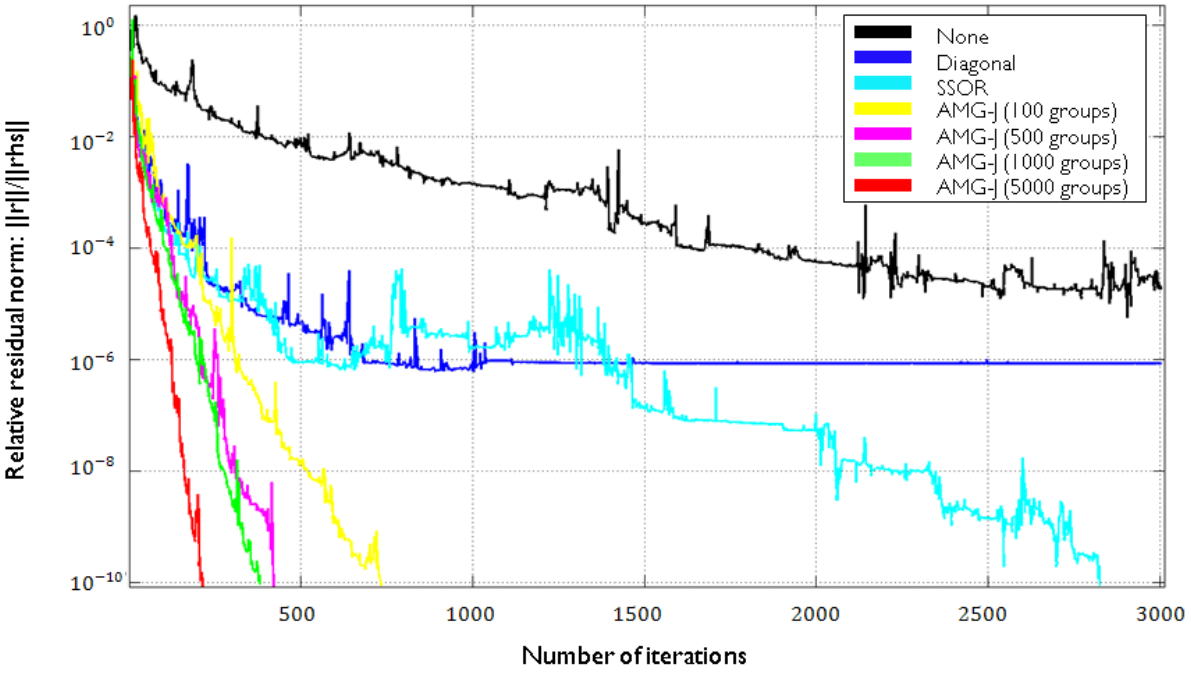


Figure 4.12: Convergence of BiCGStab without preconditioning (black), with diagonal preconditioning (blue), with SSOR preconditioning whose over-relaxation parameter is 0.1 (cyan) and with AMG-J preconditioning with 1+1 basic iterations and 100 groups (yellow), 500 groups (magenta), 1,000 groups (green), 5,000 groups (red), for the two-layer model with the small conductivity contrast.

Fig. 4.12 clearly shows that the BiCGStab solver without any preconditioning cannot reach the prescribed precision after 3,000 iterations. It has the same problem when using only simple Jacobi preconditioning. If the solver employs SSOR preconditioning, it converges after 2,823 iterations and the execution time of the code is 145.04 s. The results obtained with AMG

preconditioning are given in Table 4.1. It is obvious that, for this model, the presented preconditioner improves convergence of the solver, as well as execution time of the code to a great extent and for all the parameter configurations used. Namely, compared to SSOR preconditioning, AMG can reduce the number of iterations by two orders of magnitude and execution time by one order of magnitude.

Big Conductivity Contrast

Next, I consider the case when the model has a big conductivity contrast, $\sigma_1/\sigma_0 = 10^5$, as defined in Subsection 4.1.1. The results have shown that AMG-J preconditioning cannot help the solver to converge to the desired precision in this case. Not even when using 3+3 basic iterations and 10,000 groups. However, the SSOR smoothing scheme can improve the convergence so that the solver can reach the expected precision. In order to provide a good convergence, the solver needs to employ AMG-SSOR preconditioning either with 3+3 basic iterations and at least 500 groups, or with 1+1 basic iterations and at least 1,000 groups. The conclusion is that this model with high conductivity contrast is more challenging and, hence, requires more expensive versions of the AMG preconditioner. Fig. 4.13 shows the convergence reached when the BiCGStab does not use any preconditioner and when it uses different preconditioning schemes, including the AMG-SSOR preconditioner with 3+3 basic iterations and a varying number of groups.

The chart in Fig. 4.13 shows that, in this case, BiCGStab reaches the prescribed precision in less than 3,000 iterations only when preconditioned with the AMG-SSOR preconditioner that has 3+3 basic iterations and 500 or more groups. When there are 500 groups, the solver converges in 2,060 iterations and the execution time is 130.63 s. For 1,000 groups, the convergence is reached after 1,303 iterations and the execution time is the shortest achieved for this case: 94.37 s. And for 5,000 groups, the solver needs only 781 iterations to reach the desired precision, while the code needs 179.63 s to finish execution.

4.3.2 Seven-Material Model

The second model, presented in Fig. 4.14, is a completely artificial one, made with the sole purpose of testing the preconditioning scheme for quite a complex conductivity structure featuring extremely large conductivity contrasts. In this way, I want to simulate what may appear in real geological structures and what is usually a source of numerical problems. The values of conductivities are: $\sigma_1 = 0.1$ S/m, $\sigma_2 = 1.0$ S/m, $\sigma_3 = 10.0$ S/m, $\sigma_4 = 100.0$ S/m, $\sigma_5 = 1000.0$ S/m, $\sigma_6 = 50.0$ S/m, $\sigma_7 = 500.0$ S/m. Clearly, the conductivity contrasts vary from 10 up to 10^4. The mesh and the source are the same as in the two-layer model.

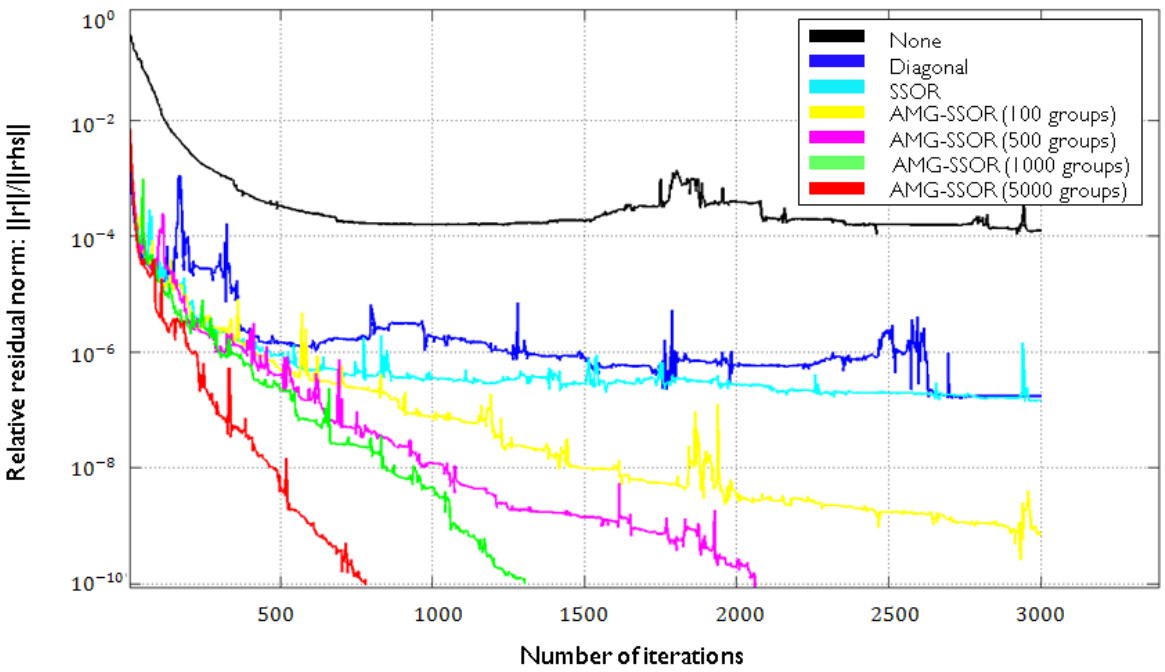


Figure 4.13: Convergence of BiCGStab without preconditioning (black), with diagonal preconditioning (blue), with SSOR preconditioning whose over-relaxation parameter is 0.1 (cyan) and with AMG-SSOR preconditioning with 3+3 basic iterations and 100 groups (yellow), 500 groups (magenta), 1,000 groups (green), 5,000 groups (red), for the two-layer model with the high conductivity contrast.

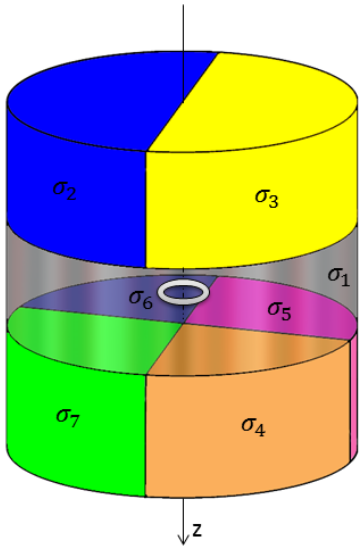


Figure 4.14: Model of a conductivity structure composed of seven different conductive materials with a horizontal finite current loop as the source.

This model has proved to be quite difficult for all the tested preconditioning techniques. As expected, taking into account the results for the two-layer model with the high conductivity contrast, the AMG-J preconditioner is rather helpless in this case, so that AMG-SSOR is the only scheme that helps the solver to converge. However, in order to achieve the desired convergence, it is necessary to employ very expensive versions of the AMG-SSOR preconditioner. Namely, the proper convergence is reached only when using at least 5,000 groups. With 3+3 basic iterations and 5,000 groups, the convergence is reached in 1,650 iterations, while execution time is 359.89 s. For 10,000 groups, the solver converges after 933 iterations and the code is executed in 495.35 s. Fig. 4.15 presents the solver's convergences for different preconditioning schemes, including the AMG-SSOR preconditioner with 3+3 basic iterations, which is more efficient than the 1+1 AMG-SSOR version, and different number of groups.

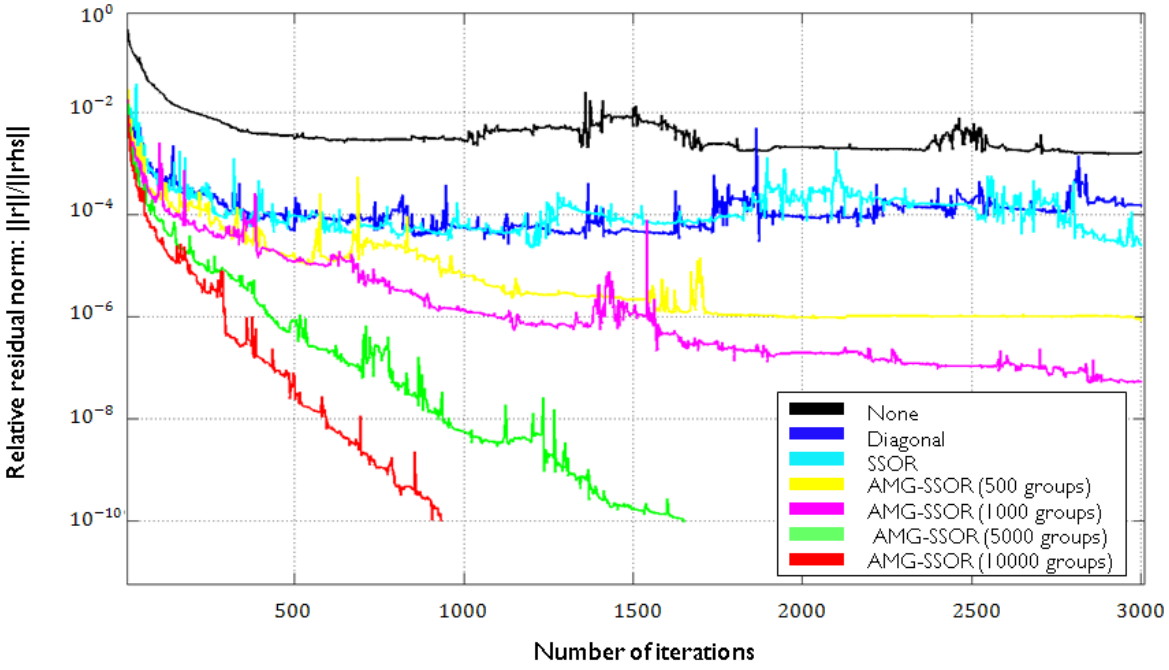


Figure 4.15: Convergence of BiCGStab without preconditioning (black), with diagonal preconditioning (blue), with SSOR preconditioning whose over-relaxation parameter is 0.1 (cyan) and with AMG-SSOR preconditioning with 3+3 basic iterations and 500 groups (yellow), 1,000 groups (magenta), 5,000 groups (green), 10,000 groups (red), for the seven-material model.

4.3.3 Flat Seabed Model

The next numerical example is the model described in Subsection 4.1.2. Since this model is not very challenging numerically, employment of any of the tested preconditioners can significantly

improve the convergence of the solver so that it is able to give a suitable approximation rather easily. Therefore, the question for the proposed AMG preconditioning is how much faster it can be compared to the other schemes that are less demanding in terms of computational requirements inside of one solver iteration. Considering that any combination of AMG parameters can ensure a good convergence, the best idea is to use the computationally cheapest parameters. Taking this into account, I compare the convergences obtained using AMG-J with 1+1 basic iterations and different number of groups (up to 5,000) with the convergences generated by the other schemes, Fig. 4.16.

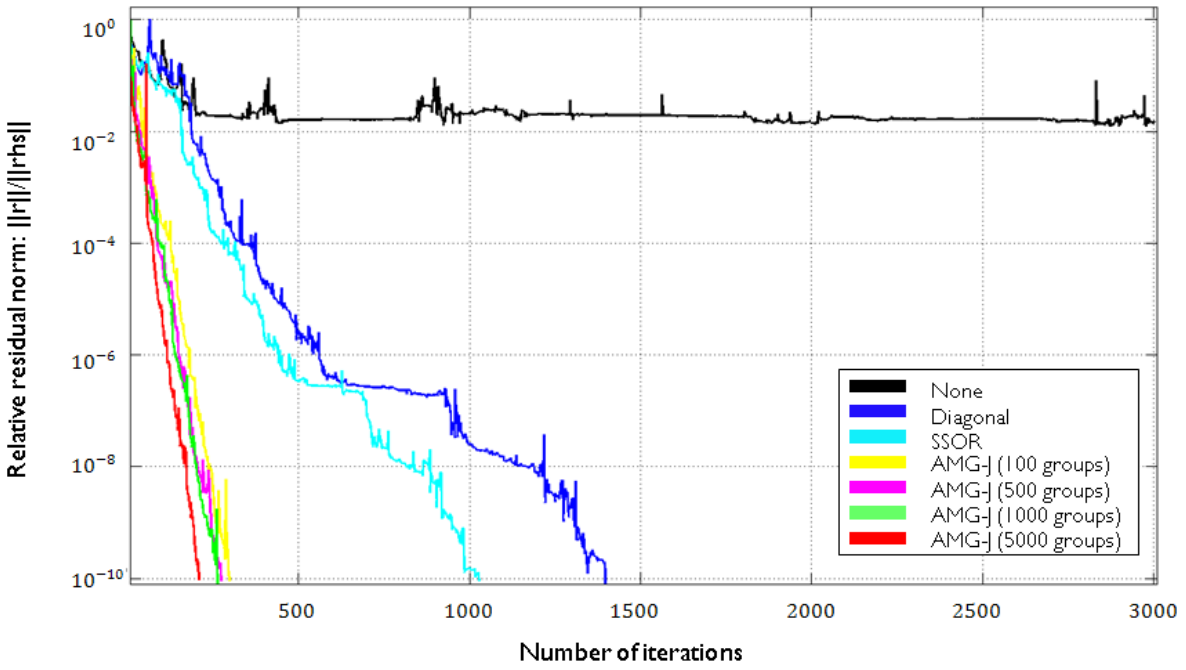


Figure 4.16: Convergence of BiCGStab without preconditioning (black), with diagonal preconditioning (blue), with SSOR preconditioning whose over-relaxation parameter is 0.1 (cyan) and with AMG-J preconditioning with 1+1 basic iterations and 100 groups (yellow), 500 groups (magenta), 1,000 groups (green), 5,000 groups (red), for the flat seabed model.

Although AMG preconditioning greatly improves the convergence of the solver, the gain in execution time of the code is not so spectacular. This is demonstrated in Table 4.2.

The values in the table show that AMG can reduce the number of iterations six times, compared to the SSOR preconditioner, or seven times, compared to diagonal preconditioning. On the other hand, the biggest reduction in execution time is around four times, compared to SSOR. And compared to diagonal preconditioning, the execution time can be reduced only 1.3 times. Although it is not significant, clearly there is some improvement obtained by employing

Preconditioner	Diagonal	SSOR	AMG with 1+1 Jacobi & 100 groups	AMG with 1+1 Jacobi & 500 groups	AMG with 1+1 Jacobi & 1000 groups	AMG with 1+1 Jacobi & 5000 groups
Convergence (No. of iterations)	1397	1273	298	274	264	209
Total execution time (sec.)	55.21	159.35	45.58	43.25	44.97	80.61

Table 4.2: Results for the flat seabed model.

AMG preconditioning for this model, as well.

4.3.4 Tests for Insensitivity to the Maximal Size Ratio Between the Grid Elements

FE grids normally have extremely large size ratios between the elements due to local refinements. This is usually a reason for the poor convergence of a solver (Koldan *et al.*, 2011). Having this in mind, I have performed tests to see if this preconditioning scheme can reduce the sensitivity of an iterative solver to a big size difference between the biggest to the smallest element in a mesh.

Two meshes of almost the same sizes have been created, but with very different local refinements and hence significantly different ratios between the sizes of the elements for the canonical disc model described in Subsection 4.1.3.

First mesh is quasi-uniform since it has a very small and simple refinement: the size of the largest element is only two times bigger than the size of the smallest one. It is quite easy to create such a mesh because it is not necessary to put much effort into the refinement process. This mesh has 2,993,420 elements and 512,060 nodes (2,048,240 degrees of freedom). Although with very simple refinement, this mesh has proved to have enough elements to provide a sufficiently accurate solution approximation in this case. Namely, the quality of approximations to the EM field vectors is almost the same for both meshes.

The results for this mesh have shown that the solver converges to the given precision with any of the tested preconditioning schemes. Fig. 4.17 shows that the AMG-J preconditioner with 1+1 basic iterations and a variant number of groups performs much better than other preconditioners. It reduces the number of iterations up to 11 times compared to the diagonal preconditioner, and up to 8 times when compared to SSOR.

When comparing execution times, Table 4.3 shows that AMG-J can be almost 2 times faster than diagonal preconditioning, and 5.5 times faster than SSOR.

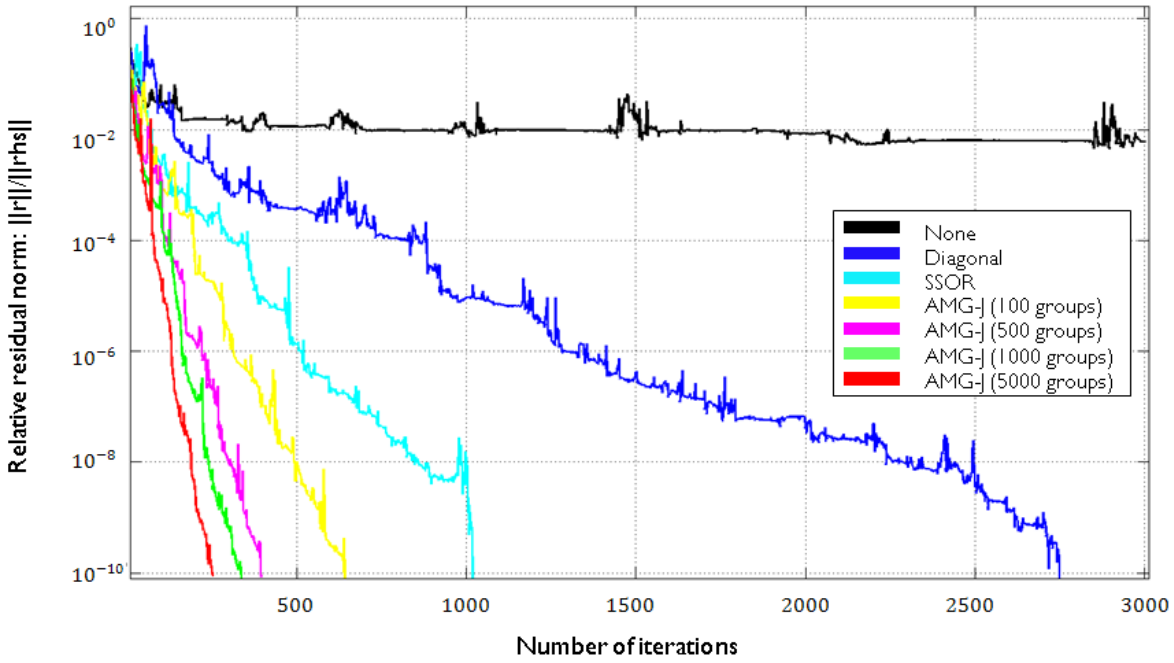


Figure 4.17: Convergence of BiCGStab without preconditioning (black), with diagonal preconditioning (blue), with SSOR preconditioning whose over-relaxation parameter is 0.1 (cyan) and with AMG-J preconditioning with 1+1 basic iterations and 100 groups (yellow), 500 groups (magenta), 1,000 groups (green), 5,000 groups (red), for the quasi-uniform mesh for the canonical disc model.

Preconditioner	Diagonal	SSOR	AMG with 1+1 Jacobi & 100 groups	AMG with 1+1 Jacobi & 500 groups	AMG with 1+1 Jacobi & 1000 groups	AMG with 1+1 Jacobi & 5000 groups
Total execution time (sec.)	103.92	306.89	96.74	64.19	55.25	84.09

Table 4.3: Results for the quasi-uniform mesh for the canonical disk model.

The second mesh used for simulations greatly exploits the power of FE method having huge local refinements around the source and receivers, as well as in the centre of the model. The ratio of the size of the biggest element to the size of the smallest one is 100:1. However, it is necessary to have a powerful mesh generator to create a high-quality mesh with such big refinements. This mesh has 2,991,478 elements and 511,020 nodes, which means 2,044,080 degrees of freedom.

The results for the second mesh have shown that convergence to the desired precision can be reached only when using AMG preconditioning. Since any combination of the tested AMG parameters gives good convergence, the computationally least demanding version of the preconditioner should be chosen. Therefore, in Fig. 4.18, where is given the comparison of the solver's performances when preconditioned with different schemes, I present the results obtained by AMG-J with 1+1 basic iterations and variant number of groups. The best execution time of 45.35 s is gained when using 1+1 Jacobi iterations and 1,000 groups.

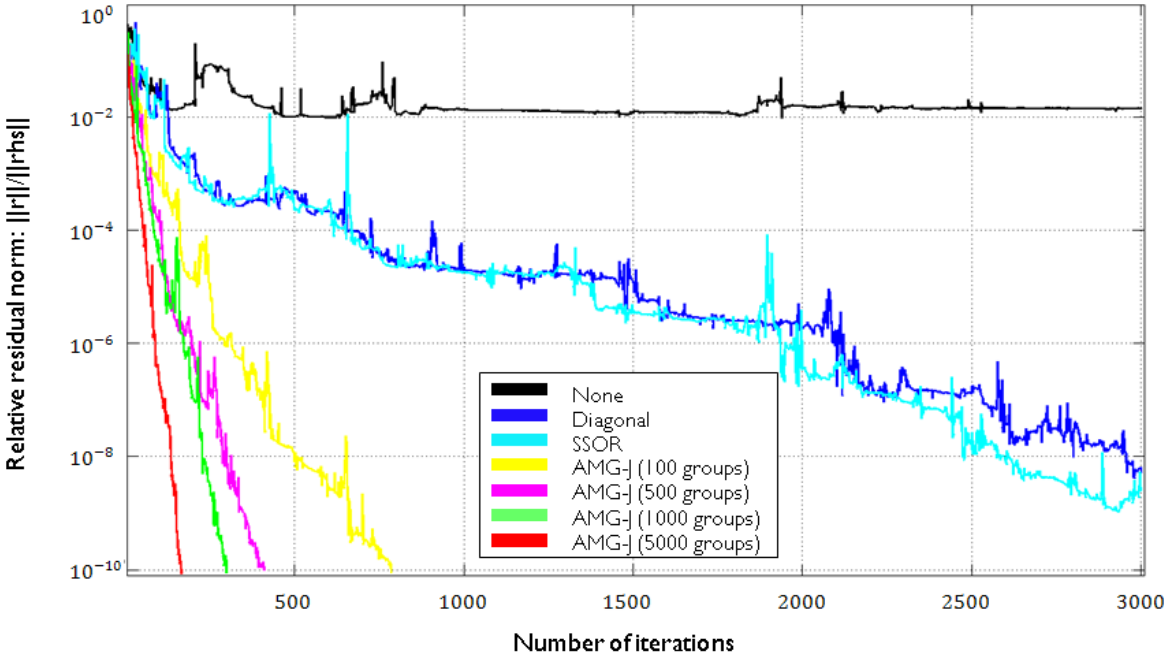


Figure 4.18: Convergence of BiCGStab without preconditioning (black), with diagonal preconditioning (blue), with SSOR preconditioning whose over-relaxation parameter is 0.1 (cyan) and with AMG-J preconditioning with 1+1 basic iterations and 100 groups (yellow), 500 groups (magenta), 1,000 groups (green), 5,000 groups (red), for the very refined mesh for the canonical disc model.

Generally, any local refinement of a mesh produces a deterioration in a solver's convergence. However, this set of tests has demonstrated that the AMG preconditioner can considerably improve the convergence of the solver no matter how big local refinements of the mesh are.

Furthermore, if the results in Fig. 4.17 and Fig. 4.18 are compared, it can be seen that the solver with any version of AMG preconditioning converges to the desired precision after almost the same number of iterations for both meshes. It may be concluded that the AMG preconditioned solver is quite insensitive to the maximal size ratio between grid elements.

4.3.5 Tests for Grid-Independent Rate of Convergence

In order to prove a grid-independent rate of convergence of the BiCGStab solver when preconditioned with AMG, I have performed experiments using the automatic mesh refinement that is built in *Alya* code to create larger meshes. As already described in Chapter 2, Subsection 2.2, at each subsequent level of the refinement, each tetrahedron of the current mesh is divided into 8 new tetrahedra. Clearly, this leads to a new mesh that is 8 times bigger than the preceding one. Due to the size of the new mesh, it is necessary to create 8 times more groups than for the lower-level mesh. In this way, the relative reduction of the fine-space dimension, i.e. dimension of the linear system, to a coarse-space dimension stays the same.

In all tests, I have performed automatic refinement up to the second level because of the enormous sizes of the meshes. This means that in each case there are three meshes for comparison, which is enough to show how the convergence of the solver preconditioned with AMG is (un)affected by the increase in number of elements, i.e. number of unknowns. The results of these tests for the two-layer model, flat seabed model and canonical disc model are given in Tables 4.4, 4.5 and 4.6, respectively.

Level of refinement	Number of elements (10^6)	Number of nodes (10^3)	Number of iterations	Time of execution (sec.)
0	0.5	93.4	1,309	75.95
1	4.3	734.3	1,148	505.08
2	34.8	5,800.0	1,063	3,731.32

Table 4.4: Convergence and execution time for different refinement levels of the mesh used for the two-layer model.

It can be observed that the convergence is quite independent of the mesh size in all cases. This means that the scheme really does guarantee convergence for extremely large meshes ($\sim$ 200 million elements). However, while the number of iterations almost does not change with the size, the execution time grows nearly linearly. This is due to the fact that for bigger meshes it is necessary to create more groups, which means a bigger coarse system and more time for its factorisation and solution. Because of this drawback of the scheme, it is preferable to use as few groups as possible for the original mesh. In the presented tests, I have used 100 groups for

Level of refinement	Number of elements (10^6)	Number of nodes (10^6)	Number of iterations	Time of execution (sec.)
0	3.1	0.5	476	41.40
1	25.0	4.2	514	333.61
2	199.8	33.5	597	3,049.46

Table 4.5: Convergence and execution time for different refinement levels of the mesh used for the flat seabed model.

Level of refinement	Number of elements (10^6)	Number of nodes (10^6)	Number of iterations	Time of execution (sec.)
0	2.99	0.5	1,812	145.42
1	23.9	4.0	1,897	1,147.84
2	191.5	32.0	1,764	8,302.98

Table 4.6: Convergence and execution time for different refinement levels of the mesh used for the canonical disc model.

the first mesh. Consequently, 800 groups have been created for the first and 6,400 groups for the second level of refinement.

4.3.6 Complex Real-Life Synthetic Model

The presented AMG preconditioning scheme has been also tested for the model described in Subsection 4.1.4. In these tests, the convergence criterion for all BiCGStab iterations is a reduction of the relative residual norm to a value in the order of 10^{-8}, while the number of iterations has been limited by the maximum value of 1,000. In Fig. 4.19, the convergence of the solver for different preconditioning schemes is compared. Fig. 4.19 shows that BiCGStab converges to the desired precision in less than 1,000 iterations only when using AMG preconditioning. The results have shown that any reasonable combination of AMG parameters gives good convergence, so, Fig. 4.19 presents the results obtained by AMG-J with 1+1 basic iterations, which is computationally the least demanding version of the preconditioner. For this case, the shortest total execution time is 37 s.

4.3.7 Discussion

A series of experiments for several models with different characteristics have been performed to test the performance of the proposed AMG preconditioning technique when combined with the BiCGStab method. The results have shown that the AMG preconditioner greatly improves the

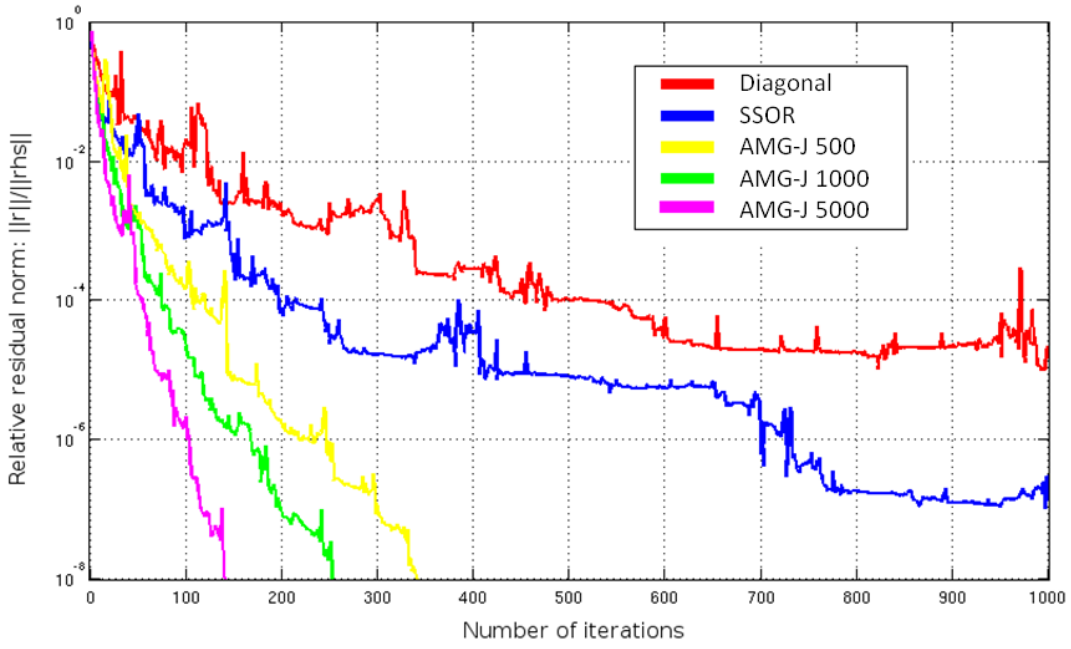


Figure 4.19: Convergence of BiCGStab with different preconditioners for the original problem (2 million degrees of freedom) for the complex real-life synthetic model.

convergence of the solver for all tested cases. The convergence becomes better with the increase of the number of basic iterations, as well as the size of the coarse system. However, these increases introduce additional computational costs that slow down the execution of a single iteration. Therefore, it is important to carefully find the right balance of all the parameters in order to obtain the best possible performance. The choice of parameters is not a trivial task and there are no straightforward rules for it. However, I have drawn some conclusions from the tests I have performed that can help to create a rough general strategy for choosing AMG parameters. The experiments have shown that, in most cases, the best results are achieved when using only 1+1 Jacobi iterations and $500 - 1,000$ groups. But, if a model is very complex and has high conductivity contrasts, it will be probably necessary to create up to 5,000 groups and to have more basic iterations. Similarly, SSOR relaxations have proved to be more efficient in dealing with high contrasts between conductivities than Jacobi iterations. In the described examples, the systems that have been solved have between 0.5 and 2 million unknowns, which means that numbers of groups that proved to be the most efficient choices are three or four orders of magnitude smaller. I remark that these ratios may be used as guidance when choosing the number of groups.

Compared to other preconditioning schemes, such as diagonal and SSOR, the AMG preconditioner has proved to be especially useful in cases with big conductivity contrasts, high

frequencies employed and/or large maximal size ratio between the mesh elements. In these situations, in which the other preconditioners have problems to ensure the desired convergence, there is always at least one version of AMG preconditioned solver that is able to converge to the prescribed precision in less than 3,000 iterations. Furthermore, in situations when other preconditioning techniques work very well, computationally cheap versions of the AMG preconditioner can improve the performance of the solver even more. Namely, despite the extra cost per iteration, if the right combination of parameters is chosen, AMG is always able to reduce the solution time compared to the other preconditioning schemes.

Tests have shown that the presented AMG preconditioner ensures the convergence of a Krylov subspace method which does not depend on the size of a mesh. This means that one can increase the size of a grid and the solver will converge after almost the same number of iterations. However, for a bigger mesh it is necessary to create a larger number of groups, which introduces an additional computational overhead that increases the execution time almost linearly. This drawback could be overcome to some extent by parallelisation of the AMG procedure. In addition to this, it has been shown that the preconditioner improves both the convergence and the execution time for simple (quasi-)uniform meshes as well as for complex ones with huge local refinements. Moreover, the convergence of the AMG preconditioned solver is rather unaffected by the ratio between the sizes of the grid elements.

AMG has been implemented as a black-box preconditioner, so that it can be easily used and combined with different iterative methods. However, this has not been completely obtained yet since a user has to choose AMG parameters in order to achieve the best performance. For having a perfect black-box preconditioner, it is necessary to build in the code an automatic way of choosing optimal AMG parameters. One of possible ways to do this is to employ a suitable machine learning algorithm.

4.4 Scalability Tests

4.4.1 Scalability Using MPI Communication

The scalability of the code when using only MPI communication has been tested by running the same problem for different numbers of CPUs working in parallel. I have measured the total time spent on the construction of the system matrix and RHS vector and on solving the system. The first part takes approximately 1% of the total time, which means that most of the CPU usage goes to the linear-system solver. All simulations have been carried out on MareNostrumIII supercomputer (**http://www.bsc.es**).

MareNostrumIII is a supercomputer based on Intel SandyBridge processors, iDataPlex Compute Racks, a Linux Operating System and an Infiniband interconnection. Its peak performance is 1 Petaflops. It has 48,448 Intel SandyBridge-EP E5-2670 cores with a frequency of 2.6 GHz, grouped into 3,028 computing nodes, 94.625 TB of main memory (32 GB/node), 1.9 PB of disk storage as well as infiniband and gigabit ethernet interconnection networks. The computing nodes are the last generation of IBM System X servers: iDataPlex dx360 M4. These nodes are based on Intel Xeon (R) technology, and they offer high performance, flexibility and power efficiency. Each computing node has two 8-core Intel Xeon processors E5-2670 with a frequency of 2.6 GHz and 20 MB cache memory.

It is important to emphasise that all the models presented here are synthetic and hence much smaller and simpler than real ones, which cannot be presented in this work. Namely, the employed test models have approximately 3 million elements and 0.5 million nodes (2 million degrees of freedom), while real ones normally have thousands of millions of elements and tens and even hundreds of millions of nodes and degrees of freedom. Therefore, I have used the automatic mesh refinement of the second level that is built in the code to create very large meshes that are of sizes of those used in real industrial applications. In this way, the scalability of the code can be evaluated in a way that is relevant for real industrial purposes. For the following tests, I have used the realistic complex model described in Subsection 4.1.4 and the mesh created for it that has 2,996,017 elements and 512,651 nodes (2,050,604 degrees of freedom). The second level of the automatic mesh refinement has created a big mesh that has 191.7 million elements and 32.1 million nodes (128.4 million degrees of freedom), which is close to sizes of real industrial meshes. Simulations for this big second-level mesh require at least 32 slaves due to its huge memory demands. Therefore, these tests have been performed on 32, 64, 128, 256, 512 and 1024 CPUs, using all 16 cores per node.

Synchronous MPI Communication

First set of tests has been carried out using synchronous MPI communication described in Chapter 3, Subsection 3.2.1.

Fig. 4.21 shows speed-ups obtained with synchronous MPI communication (marked with triangles) for up to 1024 CPUs for the second-level mesh (191.7 million elements and 128.4 million degrees of freedom).

The achieved scalability is almost linear for up to 256 processors. From this number on, the scalability stops its near-linear growth and slowly begins to saturate since the execution becomes dominated by exchange of messages between the processes. However, the speed-ups

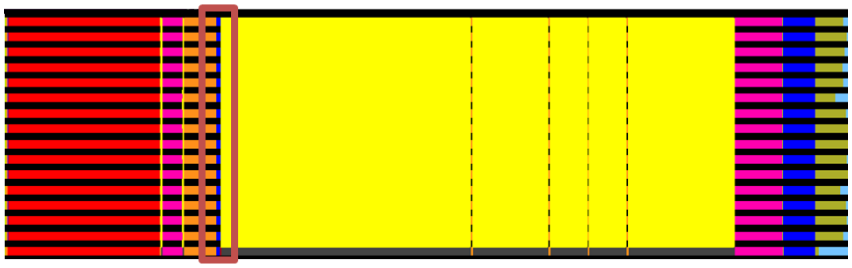
keep growing constantly and significant improvements in execution time for more than thousand processors has been observed.

In order to perform a more thorough analysis of the parallelisation with synchronous MPI, I have carried out another series of executions on 2, 4, 8, 16 and 32 CPUs using the original mesh that has 3 million elements. Since the mesh is rather small, all the effects that appear with the increase of the number of processes can be noticed much earlier. Fig. 4.20 shows traces of these program executions. It can be seen that matrix assembly (dark blue part framed with red line), which is small (1% of the total execution time), perfectly parallel part of the code, quickly disappears with the increase of the number of processes. However, iterative solver (yellow part), which is dominant and expensive part of the code, requires communication and therefore does not have the ideal linear scaling. Taking this into account, it is clear that the overall speed-up depends on the behaviour of the solver in the parallel context. Tables 4.7 – 4.11 give more insights into what happens when the number of CPUs is increased and explain why the speed-up starts saturating at some point of this increase. These tables present the minimal, maximal and average time that processes have spent on performing useful computations, i.e. running (dark blue colour in Fig. 4.20). Also, they show the minimal, maximal and average time that has been spent on communication – group communication with `MPI_Allreduce` (orange parts in Fig. 4.20) as well as inter-neighbouring communication with `MPI_Sendrecv` (yellow colour in Fig. 4.20). Finally, they demonstrate the percentages of time that each of these events have taken within the execution. Analysing these results, it is easy to notice that the computation time has been reduced by increasing the number of processes – around 8 times when increasing the number of CPUs from 2 to 32. On the other hand, the time spent on communication has been increased with the number of used CPUs – around 4.5 times when increasing the number of processes from 2 to 32. In other words, when using 2 CPUs, computation takes 98% of the overall time and communication only 2%, while when using 32 CPUs, communication becomes dominant with 52% of the overall time compared to 48% that goes to computation. It is clear that communication becomes dominant with the increase of the number of processes, because of which the speed-up begins to saturate and moves away from the linear curve. Also, it can be seen that the difference between minimal and maximal computation time is smaller for fewer processes. If there are more processes, this difference becomes larger. This observation indicates that it is more difficult to create well-balanced partitions for a bigger number of CPUs. Due to this load-balance difference, the scalability of the computational part becomes worse when having more processes. However, this effect is never too large and therefore it is not a crucial factor for deterioration of the scalability. The main problem lies in the growing communication costs.

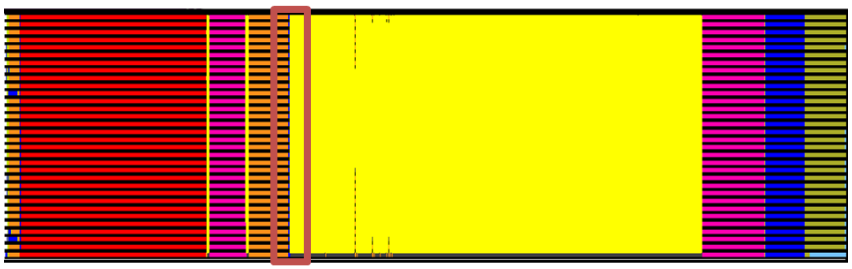
(a) Traces of 2 MPI processes.

(b) Traces of 4 MPI processes.

(c) Traces of 8 MPI processes.



(d) Traces of 16 MPI processes.



(e) Traces of 32 MPI processes.

Figure 4.20: Traces of processes with synchronous MPI communication.

Time	Running	Group communication	Send/Receive
Minimal (s)	305.051	5.249	0.513
Maximal (s)	305.053	5.462	0.716
Average (s)	305.052	5.356	0.614
Percentage (%)	98.01	1.76	0.23

Table 4.7: Execution analysis when using 2 CPUs and synchronous MPI communication.

Time	Running	Group communication	Send/Receive
Minimal (s)	159.363	7.599	0.587
Maximal (s)	163.617	7.896	4.57
Average (s)	161.547	7.751	2.501
Percentage (%)	92.92	4.48	2.60

Table 4.8: Execution analysis when using 4 CPUs and synchronous MPI communication.

Time	Running	Group communication	Send/Receive
Minimal (s)	90.17	5.561	1.235
Maximal (s)	109.451	10.087	18.4
Average (s)	100.245	7.053	9.299
Percentage (%)	79.35	7.31	13.34

Table 4.9: Execution analysis when using 8 CPUs and synchronous MPI communication.

Time	Running	Group communication	Send/Receive
Minimal (s)	44.423	7.833	3.006
Maximal (s)	75.082	14.074	34.486
Average (s)	68.31	10.597	11.795
Percentage (%)	60.72	11.38	27.90

Table 4.10: Execution analysis when using 16 CPUs and synchronous MPI communication.

Time	Running	Group communication	Send/Receive
Minimal (s)	24.57	9.212	4.527
Maximal (s)	43.103	21.028	25.403
Average (s)	38.633	14.158	12.441
Percentage (%)	48.14	23.49	28.37

Table 4.11: Execution analysis when using 32 CPUs and synchronous MPI communication.

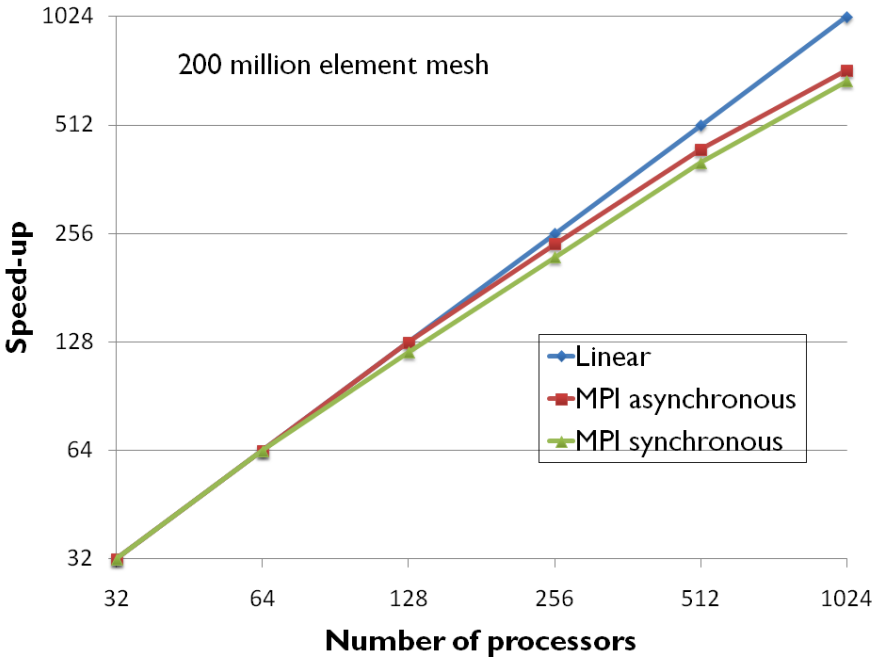


Figure 4.21: Scalability tests for the mesh with two levels of automatic refinement (128.4 million degrees of freedom) when synchronous (marked with triangles) and asynchronous (marked with squares) MPI communication is used.

Asynchronous MPI Communication

Previously, it has been shown that communication is the main source of the scalability deterioration. In order to reduce the impact of communication costs that grow with the increasing number of used CPUs, I have employed asynchronous MPI communication, described in Chapter 3, Subsection 3.2.2, which makes possible to overlap computation and inter-neighbouring communication. Here, the results of tests that have been carried out using asynchronous MPI communication are presented.

Fig. 4.21 shows speed-ups obtained with asynchronous MPI communication (marked with squares) for up to 1024 CPUs for the second-level mesh (191.7 million elements and 128.4 million degrees of freedom).

The scalability in this case behaves very similarly as the scalability in the case when synchronous communication is used. However, there is an improvement of up to 10% when asynchronous communication is employed.

In order to do a better analysis of the parallelisation with asynchronous MPI, I have carried out a series of executions on 2, 4, 8, 16 and 32 CPUs using the original mesh that has 3 million elements, like in the case of synchronous MPI communication. The results are given in Tables

4.12 – 4.16. It is clear that the influence of communication between neighbouring sub-domains has been considerably reduced. By overlapping this communication with computation, it has been obtained that bigger number of processes does not lead to the increase of time spent on inter-neighbouring communication. Moreover, this time is always almost negligible. However, the overall communication time still grows with the increase of the number of CPUs. This is due to the fact that group communication has not been reduced by asynchronous communication and therefore has become dominant communication component that grows with the increasing number of processes. Also, it stays the main source of the scalability decay. Nevertheless, the increase in communication time is considerably smaller when using asynchronous communication. For example, when increasing the number of processes from 2 to 32, communication time grows around 2 times in asynchronous case and 4.5 times with synchronous communication.

Time	Running	Group communication	Send/Receive
Minimal (s)	310.465	4.877	0.003
Maximal (s)	349.176	4.947	0.015
Average (s)	329.821	4.912	0.009
Percentage (%)	98.60	1.40	0.00

Table 4.12: Execution analysis when using 2 CPUs and asynchronous MPI communication.

Time	Running	Group communication	Send/Receive
Minimal (s)	161.261	4.948	0.005
Maximal (s)	165.069	5.302	0.009
Average (s)	162.752	5.116	0.007
Percentage (%)	96.88	3.11	0.01

Table 4.13: Execution analysis when using 4 CPUs and asynchronous MPI communication.

Time	Running	Group communication	Send/Receive
Minimal (s)	99.071	8.699	0.014
Maximal (s)	115.961	16.223	1.533
Average (s)	105.443	11.106	0.357
Percentage (%)	86.72	12.13	1.15

Table 4.14: Execution analysis when using 8 CPUs and asynchronous MPI communication.

Time	Running	Group communication	Send/Receive
Minimal (s)	45.347	7.367	0.033
Maximal (s)	73.749	11.848	0.343
Average (s)	67.927	8.944	0.138
Percentage (%)	85.81	13.79	0.40

Table 4.15: Execution analysis when using 16 CPUs and asynchronous MPI communication.

Time	Running	Group communication	Send/Receive
Minimal (s)	24.703	7.596	0.25
Maximal (s)	44.736	17.189	0.638
Average (s)	38.279	10.29	0.47
Percentage (%)	71.50	27.48	1.02

Table 4.16: Execution analysis when using 32 CPUs and asynchronous MPI communication.

4.4.2 Scalability Using Hybrid MPI–OpenMP Scheme

In order to evaluate the scalability of the code when employing the hybrid MPI–OpenMP scheme, I have performed a series of tests for a fixed number of processes and different numbers of OpenMP threads used by each process. These simulations have been carried out on **MareNostrumIII** supercomputer. Fig. 4.22 shows the results obtained with 32 processes using 1, 2, 4, 8 and 16 OpenMP threads for a case in which the coarse matrix is created from 10,000 groups and has the size of $40,000 \times 40,000$. Each process has been running on one entire node of the machine. Since each node consists of two 8-core Intel Xeon processors E5-2670, the maximal number of physical threads that can be employed in one node is 16. Fig. 4.22 also shows the maximal speed-ups that can be obtained with 2, 4, 8 and 16 threads according to Amdahl's law, which says that if P is the proportion of a program that can be made parallel and (1−P) is the rest of the code that remains serial, then the maximum speed-up that can be achieved by using N processors is:

$$\mathrm{S(N)} = \frac{1}{(1-\mathrm{P}) + \frac{\mathrm{P}}{\mathrm{N}}}. \tag{4.1}$$

When processes do not use threads, LU factorisation of the $40,000 \times 40,000$ matrix takes 75% of the execution time. According to Amdahl's law, the maximal speed-up that could be obtained for P=75% tends to 4 when N tends to infinity. Therefore, 4 is the upper limit for speed-ups in this case. Looking at Fig. 4.22, it can be seen that for up to 4 threads, the obtained speed-ups for the observed example are the same as the maximal ones. However, for a bigger number of threads, the speed-ups become smaller than their theoretical maximum. In order to

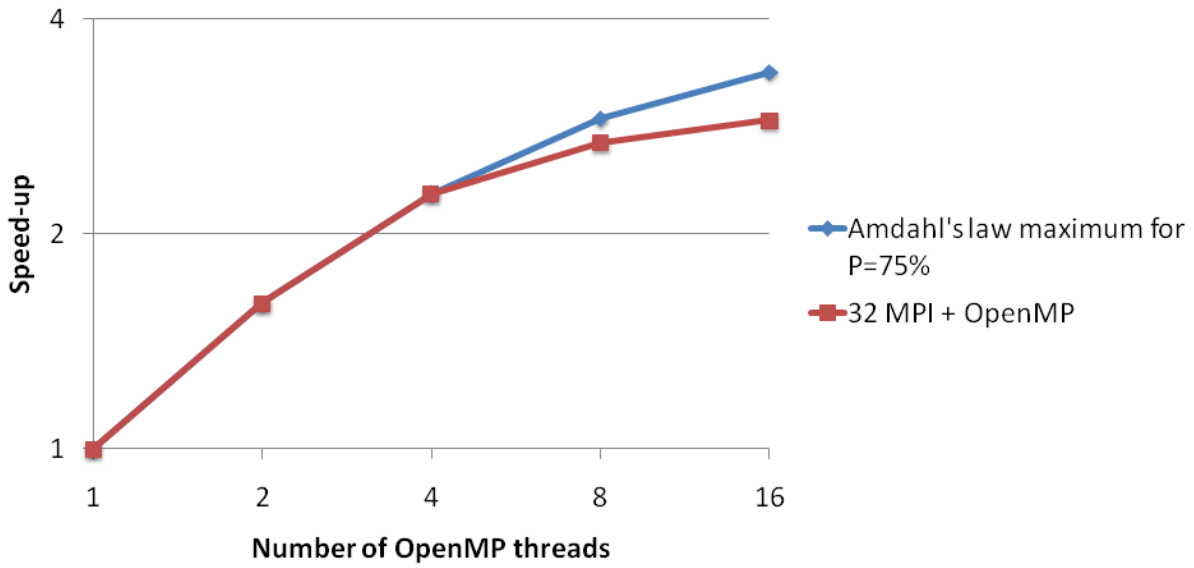


Figure 4.22: Speed-ups gained employing the hybrid MPI–OpenMP parallelisation scheme, when 32 MPI processes and different numbers of OpenMP threads within each process are used, for a case in which LU factorisation of a $40,000 \times 40,000$ matrix is performed.

understand and explain this deterioration of the overall scalability, I have tested the scalability of LU factorisation alone by running it for different numbers of OpenMP threads.

In order to test the scalability of LU factorisation, I have performed several tests for different meshes and for different sizes of coarse matrices. All simulations have been carried out on the MareNostrumIII supercomputer with 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 and 16 threads. Fig. 4.23 shows with squares the speed-ups for a coarse matrix whose size is 20,000. This coarse matrix has been created when a mesh of 3 million elements and 0.5 million nodes (2 million degrees of freedom) has been divided into 5,000 groups of nodes. With triangles, Fig. 4.23 presents the speed-ups for a coarse matrix whose size is 40,000 and that has been created from 10,000 groups generated from a mesh of 3 million elements and 0.5 million nodes (2 million degrees of freedom). Also, Fig. 4.23 shows the speed-ups for a coarse matrix whose size is 40,000 and that has been created from 10,000 groups generated from a mesh of 192 million elements and 32 million nodes (128 million degrees of freedom), which are marked with crosses.

It can be seen that in all cases the achieved speed-up is almost linear for up to 5 threads. After this number of threads, the speed-up stops its near-linear growth and slowly begins to saturate since it becomes more difficult to find enough parallelism for a bigger number of threads in elimination trees of relatively small coarse matrices that have been considered here. However,

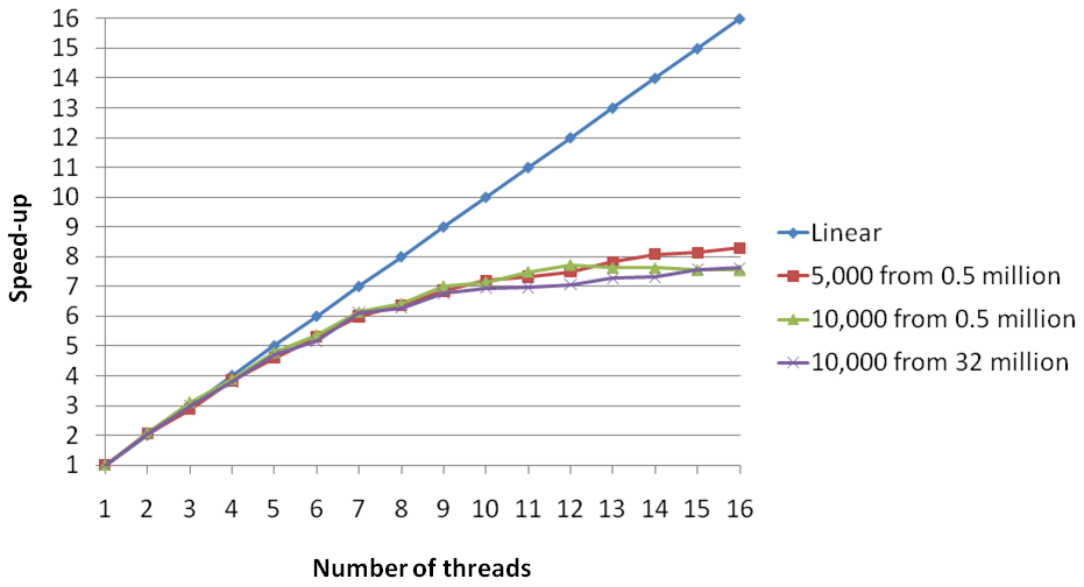


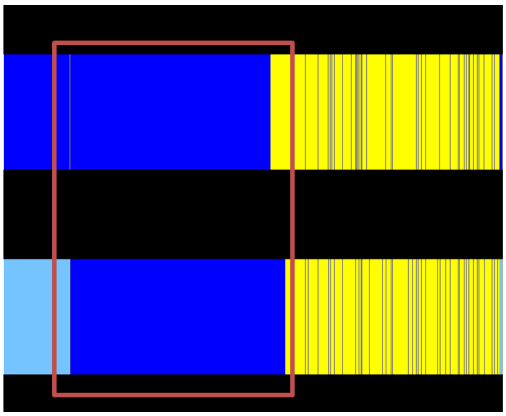
Figure 4.23: Speed-ups gained by OpenMP parallelisation of LU factorisation for: a coarse matrix created from 5,000 groups of a 0.5 million node mesh (squares), a coarse matrix created from 10,000 groups of a 0.5 million node mesh (triangles) and a coarse matrix created from 10,000 groups of a 32 million node mesh (crosses).

the speed-up keeps growing constantly and significant improvements in execution time for up to 16 threads can be observed.

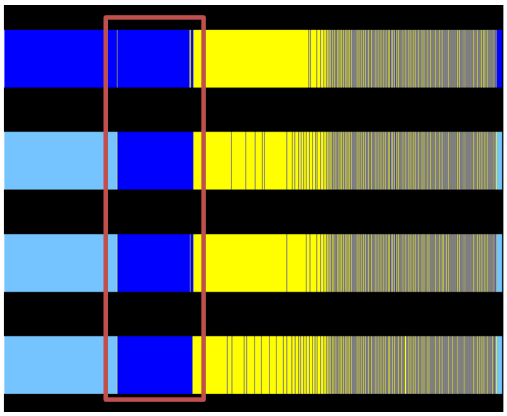
Fig. 4.24 shows traces of execution when 2, 4, 8 and 16 threads are used. Dark blue colour means that a thread is running doing some useful calculations, light blue colour marks that a thread is idle and yellow parts are scheduling and fork/join operations. Dark blue parts that are framed represent tasks that can be performed fully in parallel. The traces show that with the increase of the number of threads, these parts become smaller, so that, when 16 threads are used, these parts are extremely tiny. This demonstrates what already has been commented – it becomes more difficult to find enough parallelism in the graphs of coarse matrices for a larger number of threads.

4.4.3 Discussion

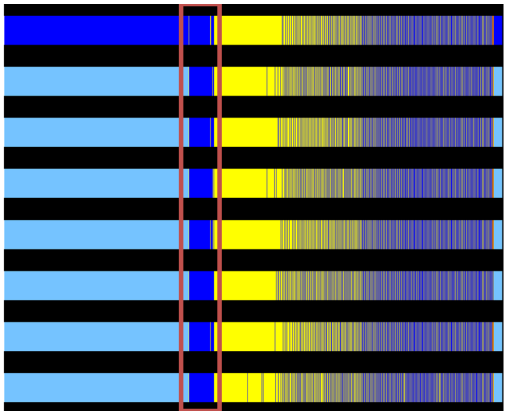
The meshes built for the models presented in this work are quite small compared to real industrial meshes. However, also for this rather small meshes, the parallelisation scheme shows very good scalability for up to 1024 CPUs. In order to evaluate what might be achieved in real applications, I have used the automatic mesh refinement in order to build really big meshes – close to ones that are used in practice and which have thousands of millions of elements and



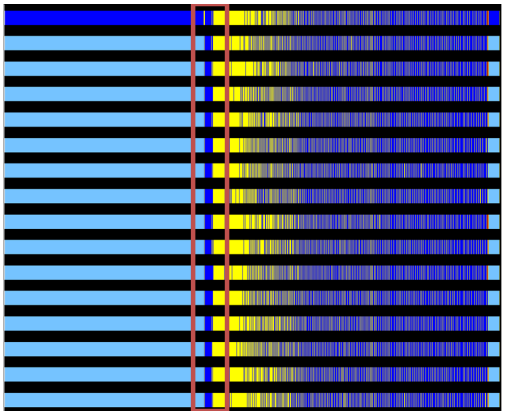
(a) Traces of 2 OpenMP threads.



(b) Traces of 4 OpenMP threads.



(c) Traces of 8 OpenMP threads.



(d) Traces of 16 OpenMP threads.

Figure 4.24: Traces of OpenMP threads used for parallel execution of LU factorisation.

hundreds of millions of nodes. Tests for these large meshes have shown extremely good scalability when using MPI communication. Up to 256 CPUs the speed-up is almost linear. After this number of processors, there is a slight decrease in speed-up due to small sub-domains and MPI communication predominance, but there is still a constant growth and quite big gain for up to 1024 CPUs. Also, the use of asynchronous MPI communication can improve the achieved speed-up up to 10%.

If an iterative solver employs AMG preconditioning, which includes time-consuming LU factorisation, it is possible to use the hybrid MPI–OpenMP parallel scheme in order to accelerate the execution of the code to a greater extent. However, the speed-up that can be obtained by OpenMP is bounded by the theoretical maximum that can be calculated according to Amdahl's law. Also, the scalability of LU factorisation alone deteriorates with number of threads, because for a bigger number of threads it is more difficult to find enough parallelism in the graphs of relatively small coarse matrices that appear in AMG.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

I have developed a nodal finite-element solver for three-dimensional electromagnetic numerical modelling in geophysics involving active sources, either current loops or arbitrary dipoles. The finite-element (FE) approach supports completely unstructured meshes, which makes the geological and bathymetric complexity of simulated models irrelevant. Thanks to this, the numerical scheme has a very broad range of applicability for real case scenarios. The employed CSEM problem formulation in terms of the secondary Coulomb-gauged electromagnetic (EM) potentials has been validated through numerous tests on different Earth models and has proved to ensure highly accurate and numerically very stable nodal finite-element solutions. Also, the presented method correctly supports electric anisotropy, which in many cases has a heavy impact on the inversion process. In addition, explicit and closed expressions for the primary electromagnetic potentials for dipole point sources, which are the most important source class in CSEM, have been developed and successfully employed for the accurate and rapid calculation of these potentials.

Three different Krylov subspace methods that are suitable for matrices that arise due to the FE discretisation of CSEM problems have been implemented. The tests that have been carried out in order to evaluate their convergence rates and resource requirements have shown that, for the problems under consideration, the BiCGStab method outperforms the other techniques in terms of convergence and/or memory consumption. The experiments have also shown that the iterative solvers converge rather poorly when employed alone or with diagonal preconditioning. Therefore, I have implemented a more elaborate preconditioning scheme for Krylov subspace methods to improve their performance and thus reduce the execution time of the whole numerical scheme, as well as to improve the accuracy of a solution. This new preconditioner is based

on algebraic multigrid (AMG) that uses different basic relaxation methods as smoothers and the wave-front algorithm to create groups, on which generation of coarse spaces is based. The AMG preconditioner has proved to be very well suited for systems that arise in 3-D CSEM numerical modelling. Namely, it has been shown that AMG preconditioning dramatically improves the convergence of Krylov subspace methods, even in the most difficult situations. The more challenging the problem is in terms of conductivity contrasts, element size ratios in a mesh and/or frequency, the more benefit is obtained by using this preconditioner. For all models shown, at least one version of the AMG preconditioner requires less computational time than other preconditioners – in some cases, the speed-up can reach an order of magnitude. Although there is no strict rule for obtaining an optimal AMG parameter set, the results vary mildly in all performed tests, making the AMG a quasi-black-box approach that can be attached to different iterative methods. Also, this preconditioner is rather practical in the parallel context. Therefore, the gain obtained by AMG preconditioning together with fully parallel solvers can be crucial in allowing EM inversion schemes to reach accurate solutions for complex 3-D scenarios in a reasonable time. Taking all the advantages and disadvantages into account, it may be concluded that, for relatively modest programming effort, I have obtained a powerful tool that can greatly improve the performance of an EM modelling scheme, which is critical for pushing EM methods towards a more common use in industry.

The presented numerical method for 3-D CSEM modelling has been implemented very efficiently on massively parallel computers. Thanks to a good parallel program design as well as the use of parallel Krylov subspace solvers and preconditioners, the scheme is able to deal with extremely large problems in a very efficient way, which makes it practical for real industrial use. The tests have shown that huge reductions in execution time can be achieved using MPI programming model – it is possible to obtain great, almost linear, scalability for more than a thousand processors. The hybrid MPI–OpenMP scheme, which has been developed for the cases in which time-consuming AMG preconditioning is used, is able to reduce the overhead introduced by AMG and thus accelerate the execution of the whole code even more. Through this work, it has been shown that there are no restrictions in employing any of the classical parallel programming models – MPI and OpenMP. Moreover, the fact that these standard programming models ensure great improvements in terms of the execution time of the code leads to the conclusion that it is enough to use these tools, without a need to search for some other up-to-date parallel programming models.

Finally, the conclusion is, what has been demonstrated with examples, that the presented parallel numerical method is very well suited to solve extremely large and complex CSEM forward problems in a very accurate, robust and highly efficient way. In addition, it is the

first 3-D CSEM solver based on finite elements that is fully parallel and highly scalable, and therefore extremely efficient.

5.2 Future Work

This thesis presents an efficient, reliable and robust numerical solution to the 3-D electromagnetic (EM) forward problem that I have successfully developed. Although this solver of the EM partial differential equations (PDE) can be used as a stand-alone solver in some applications, its main purpose is to be incorporated as a crucial part in a 3-D EM inversion algorithm, which is the final goal. Namely, in order to exploit CSEM data acquired in areas of complex geology for practical applications, it is important to create a reliable and efficient solver for the 3-D EM inverse problem. Therefore, the next development steps concentrate on creating such inversion method that is based on the modelling scheme presented in this work. The goal is to resolve the EM inverse problem using PDE-constrained optimisation algorithms that can exploit the features of the modern technologies. The discrete adjoint method, also known as 'first discretise then optimise', is a versatile and powerful technique to obtain gradients in this kind of problems. It is based on the resolution of an adjoint linear system using information from the discretised PDE. This property motivates its study and implementation using a parallel PDE solver in order to profit from the parallelisation scheme already implemented in it.

The discrete adjoint method solves the problem of finding derivatives with respect to design variables (parameters of the PDE) of a cost functional in an infinite-dimensional optimisation problem (PDE-constrained optimisation) by applying discretisation schemes in order to convert it into a finite-dimensional problem. After that, adjoint variables are spawned and the final expression of the derivatives is obtained by applying matrix-vector operations between those variables and explicit derivatives taken in the discretised constraints and the discretised cost functional. The adjoint variables are obtained by solving an adjoint problem. This problem has the same dimension of the forward problem (the discretised PDE) and in some cases uses the same left-hand side. The right-hand side is replaced by explicit derivatives of the cost functional with respect to the state variables (the main unknown in the PDE). Concerning the implementation issues of this method, as well as the fact that a very fast forward-problem code has been developed, a novel way to solve the adjoint system with the existing PDE solver, which re-uses the already assembled matrix combined with transposed parallel iterative linear system solvers, is created. Therefore, in order to compute one gradient, the following algorithm is executed for each source:

1 INPUT: γ, $\mathbf{d} := \mathbf{d}(\gamma)$

2 $\mathbf{u} \leftarrow \texttt{solve}(\mathbf{K}(\mathbf{d}), \mathbf{f}(\mathbf{d}))$

3 $\lambda \leftarrow \texttt{solve}(\mathbf{K}(\mathbf{d})^T, \overline{\mathbf{D}(\mathbf{u} - \mathbf{u}^{obs})})$

for $node_i = 1 : N$ **do**

  if $node_i \in$ *Region-Of-Interest* **then**

4 $\texttt{assemble}\left(\frac{\partial \mathbf{K}}{\partial \mathbf{d}_i}, \frac{\partial \mathbf{f}}{\partial \mathbf{d}_i}\right)$

5 $\frac{\partial \mathbf{r}}{\partial \mathbf{d}_i} = \frac{\partial \mathbf{K}}{\partial \mathbf{d}_i}\mathbf{u} - \frac{\partial \mathbf{f}}{\partial \mathbf{d}_i}$

6 $\frac{\partial j}{\partial \gamma_i} = -2\Re\left\{\overline{\lambda}^T \frac{\partial \mathbf{r}}{\partial \mathbf{d}_i}\right\} \frac{\partial \mathbf{d}_i}{\partial \gamma_i}$

  else

7 $\frac{\partial j}{\partial \gamma_i} = 0$

  end

end

8 OUTPUT: $\nabla_\gamma j(\mathbf{d})$

It can be seen that for each source (shot), it is necessary to execute the forward-problem solver two times. If we take into account that normally there are hundreds, even thousands, sources used in real industrial surveys, it is clear that there are hundreds or thousands executions of the modelling code to compute only one gradient. After obtaining one gradient, a linear search is performed, where in each step of this search, it is necessary to run the forward solver once. This linear search usually contains ten steps, which means ten more modelling code executions after computing each gradient. All in all, depending on the size of a problem and its configuration, the inversion problem can require extremely large number of executions of the forward-problem solver. Therefore, it is very important that a fast and robust solver has been developed, so that it can be efficiently incorporated in the inversion algorithm.

In addition to this, I acknowledge the fact that inverting a single data type in hydrocarbon exploration could result in ambiguities in interpretation. Therefore the goal for the future is to integrate different geophysical data sets, e.g. seismic and EM, in order to reduce the uncertainty in the characterisation of subsurface properties.

5.3 Contributions of the thesis

In the end, I would like to explicitly summarise the contributions of the presented thesis.

The main contribution of this work is a highly accurate, robust, very fast and extremely scalable numerical method for 3-D electromagnetic modelling in geophysics that is based on

finite elements and designed to run on massively parallel computing platforms. Namely, before the beginning of my research, there were no such highly efficient, fully parallel finite-element 3-D EM forward-problem solutions that would enable the development of 3-D EM inversion methods which can be used for data acquired in the environments of arbitrary geometric complexity and which are at the same time fast enough for practical use in industry. The only completely parallel and very fast code for 3-D EM modelling that existed is based on finite differences, which can lead to misinterpretations in geological environments that contain complex, irregular shapes of the structures. Now, having this new finite-element 3-D EM solver, it is possible to create a very efficient, reliable and practical 3-D interpretation method for EM data acquired in extremely complex geological environments.

In addition, this thesis discusses physical, mathematical and numerical aspects and challenges of 3-D electromagnetic modelling which have been studied during my research in order to properly design the described software for EM field simulations on 3-D areas of the Earth. Through this work, a physical problem formulation based on the secondary Coulomb-gauged EM potentials has been validated, proving that it can be successfully used with the standard nodal FE method to give highly accurate numerical solutions. Also, this work has shown that Krylov subspace iterative methods are the best solution for solving linear systems that arise after the FE discretisation of the problem under consideration. More precisely, it has been discovered empirically that the best iterative method for this kind of problems is biconjugate gradient stabilised with an elaborate preconditioner. Since most commonly used preconditioners proved to be either unable to improve the convergence of the implemented solvers to the desired extent, or impractical in the parallel context, I have proposed a preconditioning technique for Krylov methods that is based on algebraic multigrid. Different test have shown that this preconditioner is very powerful and highly effective in improving the convergence of the solvers, even in the most difficult situations, and also quite practical for parallel implementation. Finally, through this work, it has been concluded that there are not any restrictions in employing classical parallel programming models, MPI and OpenMP, for parallelisation of the presented FE solver. Moreover, these programming models have proved to be enough to provide an excellent scalability for it. Namely, the tests have shown that the achieved speed-up is close to the linear for more than a thousand processors.

Chapter 6

Publications on the Topic

Journals:

- Koldan, J, Puzyrev, V, de la Puente, J, Houzeaux, G. & Cela, J.M. Algebraic Multigrid Preconditioning within Parallel Finite-Element Solvers for 3-D Electromagnetic Modelling Problems in Geophysics. Submitted to *Geophysical Journal International* (2013).

- Puzyrev, V, Koldan, J, de la Puente, J, Houzeaux, G, Vazquez, M. & Cela, J.M. A Parallel Finite-Element Method for Three-Dimensional Controlled-Source Electromagnetic Forward Modelling. *Geophysical Journal International*, Oxford University Press (2013).

Conferences:

- Koldan, J, Puzyrev, V. & Cela, J.M. Algebraic Multigrid Preconditioner for Numerical Finite-Element Solutions of Electromagnetic Induction Problems. *SIAM Conference on Mathematical & Computational Issues in the Geosciences – GS13*, Padua, Italy (2013).

- Koldan, J, Puzyrev, V. & Cela, J.M. Algebraic Multigrid Preconditioning for Finite-Element Methods for 3-D Electromagnetic Modelling in Geophysics. *75th EAGE Conference & Exhibition incorporating SPE EUROPEC*, London, England (2013).

- Koldan, J, Puzyrev, V. & Cela, J.M. Parallel Finite-Element Method for 3-D Electromagnetic Modelling in Geophysics. *5th International Symposium on Three-Dimensional Electromagnetics – 3DEM-5*, Sapporo, Japan (2013).

- Koldan, J, Cela, J.M, de la Puente, J. & Garcia, X. Development of FE Methods for CSEM Problems and Their Application to Massively Parallel Computers. *7th International Marine Electromagnetics conference – MARELEC*, La Jolla, San Diego, CA, USA (2011).

- Puzyrev, V, Koldan, J, de la Puente, J. & Cela, J.M. Parallel Finite-Element Modeling of 3-D Electromagnetic Problems Using Potentials. *8th International Marine Electromagnetics conference – MARELEC*, Hamburg, Germany (2013).

- Puzyrev, V, Koldan, J, de la Puente, J. & Cela, J.M. A Parallel Finite-Element Approach to CSEM Forward Modeling Problems. *75th EAGE Conference & Exhibition incorporating SPE EUROPEC*, London, England (2013).

- Peredo, O, Puzyrev, V, Koldan, J, Houzeaux, G, Vazquez, M, de la Puente, J. & Cela, J.M. Inverse Modelling of 3D Controlled-Source Electromagnetics Using a Parallel Discrete Adjoint Method. *5th International Symposium on Three-Dimensional Electromagnetics – 3DEM-5*, Sapporo, Japan (2013).

Workshops:

- Koldan, J, Puzyrev, V, Cela, J.M, de la Puente, J. & Ortigosa, F. A Parallel Finite-Element Method for 3-D Marine Controlled-Source Electromagnetic Forward Modeling. *International Workshop on Gravity, Electrical & Magnetic Methods and their Applications – GEM*, Beijing, China (2011).

- Puzyrev, V, Koldan, J, de la Puente, J, Houzeaux, G. & Cela, J.M. A Massively Parallel Nodal 3D Finite-Element Approach to CSEM Problems. *AGU Fall Meeting*, San Francisco, CA, USA (2012).

References

Abubakar, A., Habashy, T., Druskin, V., Knizhnerman, L. & Alumbaugh, D. (2008). 2.5 d forward and inverse modeling for interpreting low-frequency electromagnetic measurements. *Geophysics*, **73**, F165–F177. 13

Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D. & Silva, C. (2003). Computing and rendering point set surfaces. *Visualization and Computer Graphics, IEEE Transactions on*, **9**, 3–15. 36

Alumbaugh, D. & Newman, G. (1997). Three-dimensional massively parallel electromagnetic inversionii. analysis of a crosswell electromagnetic experiment. *Geophysical Journal International*, **128**, 355–363. 6

Alumbaugh, D., Newman, G., Prevost, L. & Shadid, J. (1996). Three-dimensional wideband electromagnetic modeling on massively parallel computers. *RADIO SCIENCE-WASHINGTON-*, **31**, 1–24. 6, 13, 18, 49

Andréis, D. & MacGregor, L. (2008). Controlled-source electromagnetic sounding in shallow water: Principles and applications. *Geophysics*, **73**, F21–F32. 3

Aruliah, D. & Ascher, U. (2002). Multigrid preconditioning for Krylov methods for time-harmonic Maxwell's equations in three dimensions. *SIAM Journal on Scientific Computing*, **24**, 702–718. 16

Avdeev, D. (2005). Three-dimensional electromagnetic modelling and inversion from theory to application. *Surveys in Geophysics*, **26**, 767–799. 16

Badea, E., Everett, M., Newman, G. & Biro, O. (2001). Finite-element analysis of controlled-source electromagnetic induction using Coulomb-gauged potentials. *Geophysics*, **66**, 786–799. 14, 15, 20, 21, 36, 66

Baker, A., Jessup, E. & Manteuffel, T. (2005). A technique for accelerating the convergence of restarted GMRES. *SIAM Journal on Matrix Analysis and Applications*, **26**, 962–984. 39

Biro, O. & Preis, K. (1989). On the use of the magnetic vector potential in the finite-element analysis of three-dimensional eddy currents. *Magnetics, IEEE Transactions on*, **25**, 3145–3159. 21

Briggs, W., Henson, V. & McCormick, S. (2000). *A multigrid tutorial*, vol. 72. Society for Industrial Mathematics. 17, 42

Burnett, D. (1987). *Finite element analysis: from concepts to applications*, vol. 324. Addison-Wesley Reading, MA. 14, 19, 25, 28

Chave, A. & Cox, C. (1982). Conductivity beneath the oceans. *Journal of geophysical research*, **87**, 5327–5338. 23

Cheesman, S., Edwards, R. & Chave, A. (1987). On the theory of seafloor conductivity mapping using transient electromagnetic systems. *Geophysics*, **52**, 204–217. 2

Commer, M. & Newman, G. (2004). A parallel finite-difference approach for 3D transient electromagnetic modeling with galvanic sources. *Geophysics*, **69**, 1192–1202. 13

Commer, M. & Newman, G.A. (2008). New advances in three-dimensional controlled-source electromagnetic inversion. *Geophysical Journal International*, **172**, 513–535. 29

Commer, M., Newman, G., Carazzone, J., Dickens, T., Green, K., Wahrmund, L., Willen, D. & Shiu, J. (2008). Massively parallel electrical-conductivity imaging of hydrocarbons using the ibm blue gene/l supercomputer. *IBM Journal of Research and Development*, **52**, 93–103. 5, 6

Constable, S. (2010). Ten years of marine csem for hydrocarbon exploration. *Geophysics*, **75**, 75A67–75A81. 3

Constable, S. & Srnka, L. (2007). An introduction to marine controlled-source electromagnetic methods for hydrocarbon exploration. *Geophysics*, **72**, WA3–WA12. 3

Constable, S. & Weiss, C. (2006). Mapping thin resistors and hydrocarbons with marine EM methods, Part II–Modeling and analysis in 3D. *Geophysics*, **71**, G321–G332. 70

da Silva, N., Morgan, J., MacGregor, L. & Warner, M. (2012). A finite element multifrontal method for 3d csem modeling in the frequency domain. *Geophysics*, **77**, E101–E115. 15

Davydycheva, S. & Rykhlinski, N. (2011). Focused-source electromagnetic survey versus standard csem: 3d modeling in complex geometries. *Geophysics*, **76**, F27–F41. 13

Davydycheva, S., Druskin, V. & Habashy, T. (2003). An efficient finite-difference scheme for electromagnetic logging in 3d anisotropic inhomogeneous media. *Geophysics*, **68**, 1525–1536. 13

Druskin, V. & Knizhnerman, L. (1994). Spectral approach to solving three-dimensional maxwell's diffusion equations in the time and frequency domains. *Radio Science*, **29**, 937–953. 6

Edwards, N. (2005). Marine controlled source electromagnetics: Principles, methodologies, future commercial applications. *Surveys in Geophysics*, **26**, 675–700. 4

Eidesmo, T., Ellingsrud, S., MacGregor, L., Constable, S., Sinha, M., Johansen, S., Kong, F. & Westerdahl, H. (2002). Sea bed logging (sbl), a new method for remote and direct identification of hydrocarbon filled layers in deepwater areas. *First break*, **20**, 144–152. 3

Everett, M. (2012). Theoretical developments in electromagnetic induction geophysics with selected applications in the near surface. *Surveys in Geophysics*, **33**, 29–63. 17

Fomenko, E. & Mogi, T. (2002). A new computation method for a staggered grid of 3d em field conservative modeling. *Earth Planets and Space*, **54**, 499–510. 13

Franke, A., Börner, R. & Spitzer, K. (2007). Adaptive unstructured grid finite element simulation of two-dimensional magnetotelluric fields for arbitrary surface and seafloor topography. *Geophysical Journal International*, **171**, 71–86. 14

Freund, R. (1993). A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. *SIAM Journal on Scientific Computing*, **14**, 470–482. 39

Freund, R. & Nachtigal, N. (1991). QMR: A quasi-minimal residual method for non-Hermitian linear systems. *Numerische Mathematik*, **60**, 315–339. 36, 39

Gilbert, J.R. & Liu, J.W. (1993). Elimination structures for unsymmetric sparse lu factors. *SIAM Journal on Matrix Analysis and Applications*, **14**, 334–352. 64

Habashy, T. (2001). Rapid numerical simulation of axisymmetric single-well induction data using the extended born approximation. *Radio Science*, **36**, 1287–1306. 6

Haber, E. (2004). Quasi-Newton methods for large-scale electromagnetic inverse problems. *Inverse Problems*, **21**, 305. 16

Haber, E., Ascher, U., Oldenburg, D., Shekhtman, R. & Chen, J. (2002). 3-d frequency-domain csem inversion using unconstrained optimization. In *2002 SEG Annual Meeting*. 13

Houzeaux, G., Vázquez, M., Aubry, R. & Cela, J. (2009). A massively parallel fractional step solver for incompressible flows. *Journal of Computational Physics*, **228**, 6316–6332. 19

Houzeaux, G., de la Cruz, R., Owen, H. & Vázquez, M. (2012). Parallel uniform mesh multiplication applied to a Navier-Stokes solver. *Computers & Fluids*, **In Press**. 29

Jin, J. (2002). The finite element method in electromagnetics. 15, 34

Karypis, G. & Kumar, V. (1995). Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0. 48, 50, 51

Key, K. (2012). Marine electromagnetic studies of seafloor resources and tectonics. *Surveys in geophysics*, **33**, 135–167. 4

Key, K. & Weiss, C. (2006). Adaptive finite-element modeling using unstructured grids: The 2d magnetotelluric example. *Geophysics*, **71**, G291–G299. 14

Koldan, J., Puzyrev, V., Cela, J., de la Puente, J. & Ortigosa, F. (2011). A parallel finite-element method for 3-d marine controlled-source electromagnetic forward modeling. In *Global Meeting Abstracts, International Workshop on Gravity, Electrical & Magnetic Methods and their Applications*, 15, 12. 87

Kong, F. (2007). Hankel transform filters for dipole antenna radiation in a conductive medium. *Geophysical Prospecting*, **55**, 83–89. 23

Kong, F., Johnstad, S., Røsten, T. & Westerdahl, H. (2008). A 2.5 d finite-element-modeling difference method for marine csem modeling in stratified anisotropic media. *Geophysics*, **73**, F9–F19. 13

Li, Y. & Dai, S. (2011). Finite element modelling of marine controlled-source electromagnetic responses in two-dimensional dipping anisotropic conductivity structures. *Geophysical Journal International*, **185**, 622–636. 14

Li, Y. & Key, K. (2007). 2d marine controlled-source electromagnetic modeling: Part 1an adaptive finite-element algorithm. *Geophysics*, **72**, WA51–WA62. 3, 14

Liu, C., Mark, E., Lin, J. & Zhou, F. (2010). Modeling of seafloor exploration using electric-source frequency-domain csem and the analysis of water depth effect. *Diqiu Wuli Xuebao*, **53**, 1940–1952. 22

MacGregor, L., Sinha, M. & Constable, S. (2001). Electrical resistivity structure of the valu fa ridge, lau basin, from marine controlled-source electromagnetic sounding. *Geophysical Journal International*, **146**, 217–236. 14

Mackie, R., Smith, J. & Madden, T. (1994). Three-dimensional electromagnetic modeling using finite difference equations: The magnetotelluric example. *Radio Science*, **29**, 923–935. 13

Mukherjee, S. & Everett, M. (2011). 3d controlled-source electromagnetic edge-based finite element modeling of conductive and permeable heterogeneities. *Geophysics*, **76**, F215–F226. 15

Mulder, W. (2006). A multigrid solver for 3D electromagnetic diffusion. *Geophysical Prospecting*, **54**, 633–649. 16

Nabighian, M.N. (1987). *Electromagnetic Methods in Applied Geophysics: Volume 1, Theory*, vol. 1. SEG Books. 7

Nédélec, J. (1986). A new family of mixed finite elements in 3. *Numerische Mathematik*, **50**, 57–81. 15

Newman, G. (1995). Crosswell electromagnetic inversion using integral and differential equations. *Geophysics*, **60**, 899–911. 6

Newman, G. & Alumbaugh, D. (1997). Three-dimensional massively parallel electromagnetic inversioni. theory. *Geophysical journal international*, **128**, 345–354. xii, 6

Newman, G. & Alumbaugh, D. (2002). Three-dimensional induction logging problems, Part 2: A finite-difference solution. *Geophysics*, **67**, 484–491. 13, 16

Newman, G., Commer, M. & Carazzone, J. (2010). Imaging csem data in the presence of electrical anisotropy. *Geophysics*, **75**, F51–F61. 24

Ramamurti, R. & Löhner, R. (1996). A parallel implicit incompressible flow solver using unstructured meshes. *Computers & Fluids*, **25**, 119–132. 61

Rodi, W. & Mackie, R. (2001). Nonlinear conjugate gradients algorithm for 2-d magnetotelluric inversion. *Geophysics*, **66**, 174–187. 4

Saad, Y. (2003). *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics. 16, 37, 38, 39, 40, 41

Saad, Y. & Schultz, M. (1986). GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, **7**, 856–869. 36, 39

Schwalenberg, K. & Edwards, R. (2004). The effect of seafloor topography on magnetotelluric fields: an analytical formulation confirmed with numerical results. *Geophysical Journal International*, **159**, 607–621. 13

Schwarzbach, C., Börner, R. & Spitzer, K. (2011). Three-dimensional adaptive higher order finite element simulation for geo-electromagnetics–a marine CSEM example. *Geophysical Journal International*, **187**, 63–74. 15, 70, 71, 72, 74

Simoncini, V. & Szyld, D. (2007). Recent computational developments in Krylov subspace methods for linear systems. *Numerical Linear Algebra with Applications*, **14**, 1–59. 39, 77

Siripunvaraporn, W. & Egbert, G. (2000). An efficient data-subspace inversion method for 2-d magnetotelluric data. *Geophysics*, **65**, 791–803. 4

Smith, J. (1996). Conservative modeling of 3-d electromagnetic fields, part i: Properties and error analysis. *Geophysics*, **61**, 1308–1318. 6

Smith, J. & Booker, J. (1991). Rapid inversion of two-and three-dimensional magnetotelluric data. *Journal of Geophysical Research*, **96**, 3905–3922. 4

Streich, R. & Becken, M. (2011). Electromagnetic fields generated by finite-length wire sources: comparison with point dipole solutions. *Geophysical Prospecting*, **59**, 361–374. 22, 24

Streich, R., Becken, M. & Ritter, O. (2011). 2.5 d controlled-source em modeling with general 3d source geometries. *Geophysics*, **76**, F387–F393. 13

Stuben, K. (2001). An introduction to algebraic multigrid. *Multigrid*, 413–532. 17

Trottenberg, U., Oosterlee, C. & Schüller, A. (2001). *Multigrid*. Academic Pr. 17, 42

Tseng, H., Lee, K. & Becker, A. (2003). 3d interpretation of electromagnetic data using a modified extended born approximation. *Geophysics*, **68**, 127–137. 6

Um, E., Harris, J. & Alumbaugh, D. (2012). An iterative finite element time-domain method for simulating three-dimensional electromagnetic diffusion in earth. *Geophysical Journal International*. 15, 29

Van der Vorst, H. (1992). Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on scientific and Statistical Computing*, **13**, 631–644. 36, 39

Wang, T. & Hohmann, G. (1993). A finite-difference, time-domain solution for three-dimensional electromagnetic modeling. *Geophysics*, **58**, 797–809. 6

Wang, T. & Tripp, A. (1996). Fdtd simulation of em wave propagation in 3-d media. *Geophysics*, **61**, 110–120. 13

Ward, S. & Hohmann, G. (1988). Electromagnetic theory for geophysical applications. *Electromagnetic methods in applied geophysics*, **1**, 131–311. 23, 37, 66, 67

Weiss, C. & Newman, G. (2002). Electromagnetic induction in a fully 3-d anisotropic earth. *Geophysics*, **67**, 1104–1114. 13

Weiss, C. & Newman, G. (2003). Electromagnetic induction in a generalized 3D anisotropic earth, Part 2: The LIN preconditioner. *Geophysics*, **68**, 922–930. 16

Weitemeyer, K., Gao, G., Constable, S. & Alumbaugh, D. (2010). The practical application of 2d inversion to marine controlled-source electromagnetic data. *Geophysics*, **75**, F199–F211. 3

Xiong, Z., Raiche, A. & Sugeng, F. (2000). Efficient solution of full domain 3d electromagnetic modelling problems. *Exploration Geophysics*, **31**, 158–161. 13

Yee, K. (1966). Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *Antennas and Propagation, IEEE Transactions on*, **14**, 302–307. 6, 13

Zhang, Z. (2003). 3d resistivity mapping of airborne em data. *Geophysics*, **68**, 1896–1905. 6

Zhdanov, M., Fang, S. & Hursán, G. (2000). Electromagnetic inversion using quasi-linear approximation. *Geophysics*, **65**, 1501–1513. 6

Zyserman, F. & Santos, J. (2000). Parallel finite element algorithm with domain decomposition for three-dimensional magnetotelluric modelling. *Journal of Applied Geophysics*, **44**, 337–351. 14