The Dissertation Committee for Kyungjoo Kim
certifies that this is the approved version of the following dissertation:

Finite Element Modeling of Electromagnetic Radiation and Induced Heat Transfer in the Human Body

Committee:

Leszek F. Demkowicz, Supervisor

Victor L. Eijkhout, Co-supervisor

Robert A. van de Geijn, Co-supervisor

Jeffrey K. Bennighof

John Tinsley Oden

Ali E. Yilmaz

Finite Element Modeling of Electromagnetic Radiation and Induced Heat Transfer in the Human Body

by

Kyungjoo Kim, B.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2013

Dedicated to my family.

Acknowledgments

I am deeply in debt to my advisors, Dr. Demkowicz, Dr. Eijkhout and Dr. van de Geijn, for their guidance and support. Dr. Demkowicz led me to the research area of *hp*-finite element methods. I greatly appreciate his enthusiasm and scholarly advice, which was of great help in giving this work its present form.

Many thanks go to Dr. Eijkhout. I first met him for a research project developing a sparse direct solver. He guided me to deliver a solution by asking questions and finding logical flaws in my arguments. He also spent endless hours reading my research papers and giving me feedback, which resulted in improved versions of the papers.

I would like to thank Dr. van de Geijn for encouraging me to set my own research goal. He has carefully listened to my ideas and helped me to develop them. After having a discussion with him, I am always fully motivated to pursue my research further.

I consider myself very lucky to have been mentored by these experts in different fields. With their help, I have gained a unique skill set that can connect advanced numerical methods to high performance computing.

This research would not be acomplished without help from my colleagues. I specially thank Paolo Gatto. He has shared the burden of developing the complex three-dimensional *hp*-finite element codes by providing the geometry model-

ing package. I also acknowledge Jeffrey Zitelli, who helped debug the code and provided feedback. I thank Field van Zee for providing technical support in using `libFLAME`, and other open source library developers. Without their efforts, I would not have been able to complete this work.

Finally, I would like to thank my friends in Austin and family. They always encouraged me with their best wishes and helped me stay sane during the lonely journey in pursuing the degree.

Finite Element Modeling of Electromagnetic Radiation and Induced Heat Transfer in the Human Body

Publication No. _____

Kyungjoo Kim, Ph.D.
The University of Texas at Austin, 2013

Supervisors: Leszek F. Demkowicz
Victor L. Eijkhout
Robert A. van de Geijn

This dissertation develops adaptive *hp*-Finite Element (FE) technology and a parallel sparse direct solver enabling the accurate modeling of the absorption of Electro-Magnetic (EM) energy in the human head.

With a large and growing number of cell phone users, the adverse health effects of EM fields have raised public concerns. Most research that attempts to explain the relationship between exposure to EM fields and its harmful effects on the human body identifies temperature changes due to the EM energy as the dominant source of possible harm. The research presented here focuses on determining the temperature distribution within the human body exposed to EM fields with an emphasis on the human head. Major challenges in accurately determining the temperature changes lie in the dependence of EM material properties on the temperature. This leads to a formulation that couples the BioHeat Transfer (BHT) and Maxwell equations.

The mathematical model is formed by the time-harmonic Maxwell equations weakly coupled with the transient BHT equation. This choice of equations reflects the relevant time scales. With a mobile device operating at a single frequency, EM fields arrive at a steady-state in the micro-second range. The heat sources induced by EM fields produce a transient temperature field converging to a steady-state distribution on a time scale ranging from seconds to minutes; this necessitates the transient formulation. Since the EM material properties depend upon the temperature, the equations are fully coupled; however, the coupling is realized weakly due to the different time scales for Maxwell and BHT equations. The BHT equation is discretized in time with a time step reflecting the thermal scales. After multiple time steps, the temperature field is used to determine the EM material properties and the time-harmonic Maxwell equations are solved. The resulting heat sources are recalculated and the process continued.

Due to the weak coupling of the problems, the corresponding numerical models are established separately. The BHT equation is discretized with H^1 conforming elements, and Maxwell equations are discretized with $\mathbf{H}(\text{curl})$ conforming elements. The complexity of the human head geometry naturally leads to the use of tetrahedral elements, which are commonly employed by unstructured mesh generators. The EM domain, including the head and a radiating source, is terminated by a Perfectly Matched Layer (PML), which is discretized with prismatic elements. The use of high order elements of different shapes and discretization types has motivated the development of a general 3D hp-FE code.

In this work, we present new generic data structures and algorithms to per-

form adaptive local refinements on a hybrid mesh composed of different shaped elements. A variety of isotropic and anisotropic refinements that preserve conformity of discretization are designed. The refinement algorithms support one-irregular meshes with the constrained approximation technique. The algorithms are experimentally proven to be deadlock free.

A second contribution of this dissertation lies with a new parallel sparse direct solver that targets linear systems arising from hp-FE methods. The new solver interfaces to the hierarchy of a locally refined mesh to build an elimination ordering for the factorization that reflects the h-refinements. By following mesh refinements, not only the computation of element matrices but also their factorization is restricted to new elements and their ancestors. The solver is parallelized by exploiting two-level task parallelism: tasks are first generated from a parallel post-order tree traversal on the assembly tree; next, those tasks are further refined by using algorithms-by-blocks to gain fine-grained parallelism. The resulting fine-grained tasks are asynchronously executed after their dependencies are analyzed. This approach effectively reduces scheduling overhead and increases flexibility to handle irregular tasks. The solver outperforms the conventional general sparse direct solver for a class of problems formulated by high order FEs.

Finally, numerical results for a 3D coupled BHT with Maxwell equations are presented. The solutions of this Maxwell code have been verified using the analytic Mie series solutions. Starting with simple spherical geometry, parametric studies are conducted on realistic head models for a typical frequency band (900 MHz) of mobile phones.

Table of Contents

List of Tables

List of Figures

Chapter 1

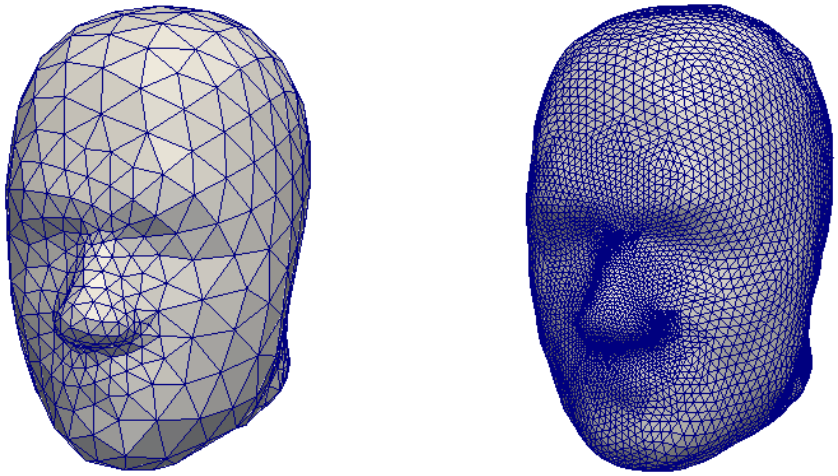
Introduction

A rapid increase in the use of wireless devices such as cell phones and wireless Local Area Networks (LANs) has produced growing concerns about the potential adverse health effects from exposure to Radio Frequency (RF), Electro-Magnetic (EM) fields emitted by those devices. This research focuses on determining the temperature distribution in the human body exposed to EM fields with an emphasis on the human head. A precise evaluation of the interaction between the human body and EM fields is important not only for the study of harmful effects on the human body but also for medical diagnoses and treatments. For example, hyperthermia treatment can shrink tumors by locally heating cancer cells with a minimum damage to neighboring normal cells. The localized deep heating inside of the human body can be obtained by means of EM radiation. The possibility of precisely determining the effects of EM radiation on the human body can ultimately be used for an optimal design of cell phones and other mobile devices to minimize adverse effects of EM waves on the tissues.

1.1 Problem statement

Coupled problem. Modeling of temperature effects of EM waves in the human body relies on the solution of Maxwell equations coupled with the Pennes BioHeat Transfer (BHT) equation. EM waves emitted from a cell phone determine the rate of heat generation in the tissue which results in a non-uniform temperature distribution in the body. In turn, the electromagnetic material properties (permittivity, permeability and conductivity) depend not only upon the radiating frequency of the antenna but also the temperature distribution in biological media. This creates a two-way coupling effect between EM waves and heat transfer procedures.

Nature of the coupling. As EM properties depend upon the operating frequency and temperature, the two models are fully coupled with each other in time, which requires discretization in both space and time. However, with an antenna operating at a single frequency, transient effects in Maxwell equations can be neglected because EM fields arrive at steady-state in micro-seconds. Hence, Maxwell equations can be approximated by time-harmonic Maxwell equations corresponding to the forcing frequency. Contrary to Maxwell equations, transient temperature changes range from seconds to minutes, and have to be resolved using the transient BHT equation. We determine the rate of heat generation from the solution of time-harmonic Maxwell equations for a single time step in thermal scales. Given the solution to time-harmonic Maxwell equations, we compute the heat sources and evolve the temperature fields by solving the BHT equation. EM material properties are updated based on the new temperature distribution, and the process continues. The



Linear model G^1-continuous model

Figure 1.1: Reconstructed G^1-continuous geometry of the human head.

two problems are never solved together, but they are weakly coupled by data transfer, in terms of variation of EM material properties and a temperature distribution, sharing the same geometry and mesh structure.

Geometry. An accurate model of the geometry of the human body is essential for the quality of solution [58, 59]. The geometry in biomedical engineering does not usually come from Computer-Aided Design (CAD) data, but it is reconstructed from an imaging system such as Computerized Tomography (CT) and/or Magnetic Resonance Imaging (MRI). We reconstruct piecewise linear geometry from MRI data using a third party software. Next, the reconstructed geometry is upgraded to a smooth, G^1-continuous representation like the one shown in Fig. 1.1 using a Geometry Modeling Package (GMP) developed by Demkowicz *et al.* [24, 98].

***hp*-Finite Element Method.** Highly accurate numerical methods are desirable to solve Maxwell equations in the complex geometry of the human head. In this dissertation, the *hp*-Finite Element Method (FEM) is used for solving the spatial and temporal distribution of temperature, and EM fields. The method uses both h and p refinements to improve the solutions: h-refinements locally reduce the element size h by subdividing the elements and p-refinements improve the solution by increasing the polynomial order p of approximation. Small elements are necessary to capture complex geometries. On the other hand, large elements with higher orders are less prone to phase (pollution) error [4, 61]. Additionally, for solutions with singularities or boundary layers, a judicious combination of h and p refinements generally achieves higher convergence rates [26, 27].

The two problems are approximated independently due to the weak coupling: the BHT equation is discretized with H^1-conforming elements while the time-harmonic Maxwell problem uses $\mathbf{H}(\mathrm{curl})$-conforming elements. Additionally, the EM domain, which includes radiating sources, must be truncated with a boundary condition that does not reflect outgoing waves but absorbs the radiating energy. To accomplish this, the Perfectly Matched Layer (PML) [12] with higher order elements is employed.

Direct solver. A large sparse system created by the FEM can be solved by either direct or iterative methods. In the *hp*-FEM, the use of direct methods is often preferred since the condition number of sparse matrices arising from the high order FEM may grow significantly with the order of approximation p [9, 17]. This

fact makes it difficult to use iterative methods without a suitable preconditioner; building a good preconditioner is another challenging task.

As a part of the dissertation, we have developed a new parallel direct solver that exploits typical sparsity patterns generated by the *hp*-FEM. In particular, the following aspects in *hp*-FEM are extensively used in the entire solution procedure:

- the linear systems generated in a sequence of consecutively adapted meshes are locally updated;
- sparse matrices derived from *hp*-discretizations consist of dense subblocks characterized by the order of approximation p.

Unlike the traditional black-box interface, the solver provides an opportunity to reuse element matrices and partial factors previously computed during the adaptive iterations [14]. Secondly, the adjacency graph of the sparse system can be constructed in a compressed form, which consists of topological nodes (*i.e.*, vertex, edge, face and interior) and their Degrees of Freedom (DOFs). This also implies that almost all matrix operations on the sparse systems based on the *hp*-mesh can be cast in terms of block matrix computations, which enable the efficient use of high performance level 3 Basic Linear Algebra Subprograms (BLAS) operations [29]. While a general sparse direct solver devotes considerable efforts to finding such supernodal information through a graph analysis, the proposed solver obtains the information from the *hp*-FEM *a priori*. These features are not yet explored by the previously developed direct solvers. Together with advances in modern High

Performance Computing (HPC) architectures, the solver outperforms conventional direct solvers for the system of equations resulting from *hp*-discretizations.

1.2 Background

In this section, we present some background on the EM wave radiation on the biological tissue and a brief review historical aspects of the numerical methods for solving Maxwell equations and the Pennes BHT model.

1.2.1 Electro-Magnetic wave radiation in biological tissues

EM waves at frequencies ranging between 1 MHz and 100 GHz, propagate through a human body, and the EM energy is absorbed by biological tissues generating localized heat sources in the medium. At the molecular level, two types of heating mechanisms are known: resistive heating by ionic conduction and dielectric heating by the inter-molecular friction of dipole molecules *e.g.*, water and proteins [45]. The energy deposited in a tissue leads to a temperature rise, which is also dependent on the cooling system of the tissue. The temperature of the tissue may continuously increase if the absorbed energy is greater than the metabolic output by the cooling system of the tissue, such as blood perfusion. As a result, the increase of temperature may cause severe damage of the tissue.

The heating pattern due to EM energy absorbed by a human body depends on various tissue properties:

- heterogeneous dielectric properties and conductivity,

- the shape of different tissues and their interfaces.

For instance, materials with high water content like muscle and brain tissue are expected to have higher energy absorption than materials with low water content such as fat and bone tissue. Besides, the complex shape of biological tissues and their interfaces may cause standing waves which result in *hot spots* [41]. This effect can lead to a deep burn of the tissue under the skin; however, the heating and damage of tissues inside of a human body are not usually noticeable because nerve cells are not evenly distributed throughout the body.

The dielectric properties and conductivity are dependent on the temperature of tissues. In the microwave spectrum, the dispersion relation is relatively small compared to the frequency dependence. The following linearized relation is given in [45]:

$$\frac{\Delta\sigma}{\sigma} = 0.02\Delta u \text{ and } \frac{\Delta\varepsilon}{\varepsilon} = -0.005\Delta u,$$

where σ and ε represent relative conductivity and permittivity, and Δu is the change in temperature with a $^{\circ}$C degree unit. Also, we note that the temperature dependence of dielectric properties of biological tissues was modeled through quadratic curve fitting for the temperature range from 20 $^{\circ}$C to 60 $^{\circ}$C in [53].

The trend of using higher frequency and higher power in mobile devices possesses increasing EM hazards [63]. When EM waves are transmitted in a human body, the waves excite specific tissues, and the tissue resistivity to the EM hazards varies based on the frequency of operation, the anatomical location, and

external environments. For example, EM heating is more dangerous to the brain and eyes than muscle tissue because the circulating blood on the muscle can distribute the heat effectively throughout the entire body. This suggests that different safety thresholds for different exposure conditions need to be established; see International Commission on Non-Ionizing Radiation Protection (ICNIRP) [2].

1.2.2 Computational ElectroMagnetics

Finite Difference Time Domain. The development of Finite Difference Time Domain (FDTD) for Maxwell equations originated in Yee's work in 1966 [101]. In his algorithm, electric and magnetic fields are approximated in space and time by using a coupled form of Maxwell equations instead of solving a single wave equation for either the electric or the magnetic field alone. The method is simple enough to deliver higher efficiency on modern parallel computing architectures. However, the method has difficulties in handling complex geometries; it requires regular structured grids and curved boundaries are approximated by staircase meshes. The method also loses accuracy when the problem involves inhomogeneous media. These bottlenecks are partially fixed by hybrid methods [11] and local mesh refinement strategies [13, 66, 104], but the revised schemes are more complicated and less efficient than those used for regular geometry.

Method of Moments. In 1968, Harrington presented a unified approach for integral equations called the Method of Moments (MOM) [42]. The method evaluates point sources that correspond to a boundary condition using a Green's function. The

basis functions satisfy the radiation condition for open domain problems. Hence, unlike the methods based on differential equations, the method does not need an artificial boundary condition to truncate a domain. Radiating fields are identified by summing up contributions from all sources. This makes the approach efficient, especially for scattering from perfect conductors; the problem reduces to an integral equation formulated on the two-dimensional scattering surface. However, the method has a drawback when the problem is formulated with inhomogeneous media. In addition, the resulting linear systems are dense, which requires intensive computations and memory usage. Many fast algorithms have been introduced to solve the problem more efficiently (*e.g.*, Fast Fourier Transform [102, 103], Fast Multipole Method [90], and Hierarchical Matrix [39] methods).

Finite Element Method. The FEM, based on the Galerkin approach, approximates solutions to Partial Differential Equations (PDEs) by constructing piecewise polynomial basis functions spanned over on a collection of subdomains called Finite Elements (FEs). The method is superior to other methods in modeling complex geometries (using elements of all shapes), and material inhomogeneities. The first FEM application to EM problems is found in Silvester's work who studied a homogeneous waveguide problem in 1969 [88]. The method was not widely accepted until a theoretical breakthrough made by Nédélec [64, 65], who solved the problem of spurious modes and introduced a family of edge-based vector elements providing the necessary (tangential) continuity in the electric (or magnetic) fields across the inter-element boundaries. These elements are also called the $\mathbf{H}(\text{curl})$-conforming

elements. There are many implementations of the $\mathbf{H}(\text{curl})$-conforming elements with the lowest order, but very few for 3D high order elements of all shapes [80]. This is because of the complexity of high order basis functions preserving tangential continuity and technical difficulties related to orientations of edges and faces.

Perfectly Matched Layer. For unbounded domain problems such as scattering or radiating waves, grid based methods such as FDTD and FEM need to truncate the unbounded domain with proper boundary conditions mimicking the waves at infinity. The Perfectly Matched Layer (PML) technique, which was first proposed by Berenger [12] in 1994 for solving the time dependent Maxwell equations, is most widely used to obtain accurate solutions on the domain of interest. The main idea of this technique is to surround the computational domain with an absorbing boundary layer of finite thickness, called the Perfectly Matched Layer. Waves travelling outward through this layer are changed into evanescent ones. The idea was re-interpreted as the *complex stretch* of variables by Chew and Weedon [22]. Recently, a systematic approach for constructing the Perfectly Matched Layer for all energy spaces forming the exact sequence has been proposed by Matuszyk and Demkowicz [60]. The construction is based on generalizing the pull-back maps (Piola transforms) to complex stretching.

1.2.3 Pennes BioHeat Transfer equation

The construction of a mathematical model for the BHT in the living tissue is very complex due to the following reasons:

1. The model needs to deal with complex thermal processes and their interactions: conduction, convection, radiation, metabolism, perfusion, phase change, and other factors.

2. The tissue structures are highly heterogeneous, and the material properties vary even if they are measured for the same kind of tissues. It is difficult or impossible to measure accurate material properties in living tissues.

3. Due to the complex geometry of blood vessels including their irregular tissue structures, the construction of precise discrete model for the human body is inherently difficult.

4. The BHT problem involves a wide range of length scales from micrometers (cells) to centimeters (tissue).

5. A precise heat-transfer model for the blood perfusion and metabolism increases significantly the complexity of the model.

The first mathematical model for BHT was proposed by Pennes [72] in 1948. Pennes conducted a set of experiments to measure the temperature distribution of forearms, and presented quantitative analysis of relationship between arterial blood and tissue temperature in the forearms. The Pennes model states the energy conservation law; the energy generated or deposited in the control volume reaches a thermal equilibrium with energy transferred energy through the surroundings. In particular, two major thermal effects are included: metabolism (heat source) and

blood perfusion (heat sink). The equation looks as follows:

$$\rho c \frac{\partial u}{\partial t} = \underbrace{\nabla \cdot (k \nabla u)}_{\text{heat flux}} + \underbrace{W_b c_b (u_{a0} - u)}_{\text{perfusion rate}} + \underbrace{\dot{q}_m}_{\text{metabolism}} ,$$

where

$\dot{q}_m =$ metabolic heat source,

$W_b =$ volumetric blood perfusion rate,

$c_b =$ specific heat of blood,

$u_{a0} =$ temperature of arterial blood,

$\rho, c, k =$ tissue density, specific heat, and effective thermal conductivity.

The model assumes that the metabolic heat source is uniformly distributed in the tissue. Blood is also assumed to be isotropically distributed in the tissue, and the fact that the blood flows in a certain direction is ignored. The convective interaction across the vascular wall is not included; instead, the model introduces the arterial blood temperature to compute total heat transfer with capillaries in the tissue.

This model is widely used in many application problems thanks to its simplicity. On the other hand, it has also been criticized for the same reason. Chen [21] pointed out that arterial blood equilibrates with the local tissue temperature before it reaches the capillary bed. Weinbaum [94] reported that the counter-current heat exchange between parallel arterial and venous vessels is an important factor. However, the modified models introduce a great deal of complexities, and they often require fine-scale models to describe complex vascular networks. In practice, the original Pennes model has been defended and validated with numerous experimental results [96].

1.3 Scope of the dissertation

The dissertation focuses on the development of hp-Finite Element technologies and a parallel sparse direct solver. The technological advances described above have been driven by the coupled Maxwell – BioHeat Transfer problem described earlier to which the developed technology has been applied. Numerical simulations presented in Chapter 6 illustrate the technology. These results represent an important step toward modeling the deposition of EM energy in the human head, but much additional work remains to be done before complete understanding of these phenomena is in hand.

Here, we give a short overview of the dissertation.

In Chapter 2, we describe variational formulation for the time-harmonic Maxwell equations and the transient Pennes BHT model. Two problems are independently formulated and weakly coupled through data transfer; the BHT model accounts for Specific Absorption Rates (SARs) induced by EM waves.

Chapter 3 is devoted to `hp3d`, a general *hp*-FE code, that supports exact sequence elements for solving multi-physics problems. The code includes elements of all shapes (*i.e.*, hexahedron, tetrahedron, prism, and pyramid) that enable to describe complex geometry. Of particular interest is the general mesh refinement scheme, which includes isotropic and anisotropic refinements for hybrid meshes that consist of the elements of all the available shapes. The proposed refinement strategy is experimentally shown to be deadlock free.

In Chapter 4, we discuss the Unassembled HyperMatrix (UHM) solver that

utilizes application information induced from hp-refinements. This direct solver has an ability to reuse the element matrices and partial factors previously computed during the mesh adaptation process. In this application context, higher efficiency is obtained from its unique workflow interface to the hp-code.

In Chapter 5, we explain our parallelization strategy for the developed sparse direct solver. Our particular interest lies in the multi-core architectures and investigated task-level parallelism. The solver exploits two-level tasking structures matching two-level parallelism in the multifrontal factorization. Coarse grain tasks are first generated via the post-order traversal in an assembly tree; next, the coarse grain task is decomposed into fine-grained tasks using algorithms-by-blocks. Those fine-grained tasks are asynchronously scheduled via multiple Directed Acyclic Graph (DAG) schedulers, where those schedulers are also scheduled during the parallel tree traversal. In effect, we achieve an efficient task scheduling system as if we have a global DAG of tasks. We show that our solver outperforms other state-of-the-art direct solvers such as PARDISO and MUMPS when the system of equations is based on higher polynomial orders. This approach is extended to heterogeneous multi-core architectures that are accelerated by multiple Graphic Processing Units (GPUs). A new task subdivision scheme is developed aiming at the efficient use of all computing resources.

In Chapter 6, we present numerical results of the BHT problem coupled with Maxwell equations. There, we focus on the verification aspect of the problem to demonstrate the effectiveness of the described FE technology.

Chapter 2

Formulation of the Problem

In this chapter, we formulate the boundary-value problem of interest: the time-harmonic Maxwell equations coupled with the transient Pennes BioHeat Transfer (BHT) equation.

2.1 Electromagnetics

It took almost a century from Newton's Law of Universal Gravitational Attraction published in his *Philosophical Naturalis Principia Mathematica* in 1687, to the analogous law in Electrostatics published by Coulomb in 1775, marking the beginning of electromagnetics. It took almost another century and work of such intellectual giants as Ampère, Faraday, Gauss, Lenz and many others until 1856 when a Scottish mathematician James Clerk Maxwell formulated what is known today as Maxwell equations; a proper mathematical model to describe the electromagnetic wave propagation. The present form of Maxwell equations is due to Heaviside who reformulated the work of Maxwell to his contemporaries in 1884. So, it took exactly two centuries from the development of the foundations of pure mechanics to the development of theories of electromagnetics and electromechanics.

2.1.1 Time-harmonic Maxwell equations

First-order Maxwell equations. In this work, we will restrict ourselves to the time-harmonic case only. For instance, we assume the following Ansatz for the unknown electric field

$$\mathbf{E}(\mathbf{x},t) = \Re\left(\mathbf{E}(\mathbf{x})e^{j\omega t}\right),$$

where ω is the angular frequency and $\mathbf{E}(\mathbf{x})$ is the unknown complex-valued phasor, being only a function of the position $\mathbf{x}$ and the frequency ω. Alternatively, the phasor can be identified as the Fourier transform (in time) of real-valued electric field $\mathbf{E}(\mathbf{x},t)$. Analogous formulas are assumed for all other involved quantities, and time derivatives in the transient Maxwell equations translate into the presence of $j\omega$ factor. Then, we obtain the following set of equations:

$$\nabla \times \mathbf{E} = -j\omega\mathbf{B} \qquad \text{Faraday's law,} \tag{2.1a}$$

$$\nabla \times \mathbf{H} = \mathbf{J} + j\omega\mathbf{D} \qquad \text{Ampère's law,} \tag{2.1b}$$

$$j\omega\rho + \nabla \cdot \mathbf{J} = 0 \qquad \text{conservation of charge,} \tag{2.1c}$$

accompanied with Gauss's laws:

$$\nabla \cdot \mathbf{D} = \rho \qquad \text{electric Gauss's law,} \tag{2.2a}$$

$$\nabla \cdot \mathbf{B} = 0 \qquad \text{magnetic Gauss's law,} \tag{2.2b}$$

where

$$\begin{aligned}
\mathbf{E} &= \text{electric field intensity} && [V/m],\\
\mathbf{H} &= \text{magnetic field intensity} && [A/m],\\
\mathbf{D} &= \text{electric flux density} && [C/m^2],\\
\mathbf{B} &= \text{magnetic flux density} && [Wb/m^2],\\
\mathbf{J} &= \text{electric current density} && [A/m^2],\\
\rho &= \text{electric charge density} && [C/m^3].
\end{aligned}$$

E and **H** fields are related to the corresponding fluxes **D** and **B** as well as the impressed current $\mathbf{J}^{imp}$ and free current **J** through the following constitutive laws:

$$\mathbf{D} = \varepsilon\mathbf{E}, \tag{2.3a}$$

$$\mathbf{B} = \mu\mathbf{H}, \tag{2.3b}$$

$$\mathbf{J} = \mathbf{J}^{imp} + \sigma\mathbf{E}, \tag{2.3c}$$

with three material parameters of the medium:

$$\begin{aligned}
\boldsymbol{\epsilon} &= \text{permittivity} && [F/m],\\
\boldsymbol{\mu} &= \text{permeability} && [H/m],\\
\boldsymbol{\sigma} &= \text{conductivity} && [(\Omega\cdot m)^{-1}].
\end{aligned}$$

These parameters are tensors for anisotropic media; they can be functions of both the position and frequency, and may also depend on the field intensities. For the case of lossy media, the material parameters become complex-valued. In this work, we consider isotropic media only; therefore, the material constants are scalars.

The Maxwell system of equations is overdetermined. Using the constitutive relations in (2.3) to eliminate **D**, **B**, and **J**, we end up with two vector unknowns: **E**, **H**, and one scalar unknown ρ; those unknowns have to satisfy two vector equations and three scalar equations. In this context, the two divergence equations are regarded as being dependent and often neglected in numerical computations. This, unfortunately, leads to spurious modes and inaccurate solutions in Computational ElectroMagnetics (CEM), and they can be corrected by using a correct discretization (*e.g.*, $\mathbf{H}(\text{curl})$-conforming elements) satisfying the divergence free conditions of the electric and magnetic fields implicitly.

Second-order wave equation. By combining (2.1a) and (2.1b) described in the first-order Maxwell equations, the second-order wave (curl-curl) equation can be formulated as follows:

$$\nabla \times \left(\frac{1}{\mu} \nabla \times \mathbf{E} \right) - \left(\omega^2 \varepsilon - j\omega\sigma \right) \mathbf{E} = -j\omega \mathbf{J}^{\text{imp}}. \tag{2.4}$$

The resulting equation is decoupled and expressed exclusively in either the electric field or magnetic field. The field variable used in this formulation is usually selected for the ease of handling given boundary conditions. We formulate equations with respect to the electric field, and the conversion to the magnetic field is made by using the Faraday's law in (2.1a).

2.1.2 Variational formulation

A variational formulation can be obtained by connecting the Ampère's law and Faraday's law: one of these two equations is satisfied weakly in the distri-

butional sense, and the other equation is imposed strongly. For the electric field formulation, (2.1b) is relaxed; a test function $\mathbf{F}$ is multiplied to the equation and integrated over the domain Ω. Integrating by parts, we obtain the following equation:

$$\int_{\Omega} \mathbf{H} \cdot \nabla \times \mathbf{F} d\mathbf{x} + \int_{\partial\Omega} (\mathbf{n} \times \mathbf{H}) \cdot \mathbf{F}_t dS = \int_{\Omega} \mathbf{J}^{\text{imp}} \cdot \mathbf{F} d\mathbf{x} + \int_{\Omega} (\sigma + j\omega\varepsilon) \mathbf{E} \cdot \mathbf{F} d\mathbf{x}. \quad (2.5)$$

Next, we impose the Faraday's law pointwise to remove the magnetic field $\mathbf{H}$:

$$\mathbf{H} = -\frac{1}{j\omega\mu} (\nabla \times \mathbf{E}). \quad (2.6)$$

By substituting (2.6) into the (2.5), the variational formula can be written as follows:

$$\int_{\Omega} \frac{1}{\mu} \nabla \times \mathbf{E} \cdot \nabla \times \mathbf{F} d\mathbf{x} - \int_{\Omega} \left(\omega^2 \varepsilon - j\omega\sigma\right) \mathbf{E} \cdot \mathbf{F} d\mathbf{x}$$
$$= -j\omega \int_{\Omega} \mathbf{J}^{\text{imp}} \cdot \mathbf{F} d\mathbf{x} + j\omega \int_{\partial\Omega} (\mathbf{n} \times \mathbf{H}) \cdot \mathbf{F}_t dS. \quad (2.7)$$

This variational equation can be rewritten with relative parameters using the following free space material parameters:

$$\varepsilon_0 = \text{free space permittivity} \quad (8.854 \times 10^{-12} F/m),$$
$$\mu_0 = \text{free space permeability} \quad (4.0\pi \times 10^{-7} H/m).$$

Then, relative parameters are set:

$$\varepsilon_r = \frac{\varepsilon}{\varepsilon_0}, \quad \varepsilon_\sigma = \frac{\sigma}{\omega\varepsilon_0}, \quad \mu_r = \frac{\mu}{\mu_0}, \quad k_0 = \omega\sqrt{\varepsilon_0\mu_0}, \quad Z_0 = \sqrt{\frac{\mu_0}{\varepsilon_0}}, \quad (2.8)$$

and the (2.7) is expressed with free space wave number k_0 and its free space impedance Z_0 as follows:

$$\int_{\Omega} \frac{1}{\mu_r} \nabla \times \mathbf{E} \cdot \nabla \times \mathbf{F} d\mathbf{x} - \int_{\Omega} k_0^2 (\varepsilon_r - j\varepsilon_\sigma) \mathbf{E} \cdot \mathbf{F} d\mathbf{x} = -jk_0 Z_0 \int_{\Omega} \mathbf{J}^{imp} \cdot \mathbf{F} d\mathbf{x} + jk_0 Z_0 \int_{\partial\Omega} (\mathbf{n} \times \mathbf{H}) \cdot \mathbf{F}_t dS.$$

Proper interface conditions can be derived from the integral form of Maxwell equations. At the interface between two media, the boundary conditions for the electric field are expressed in the following equation:

$$\hat{\mathbf{n}} \times (\mathbf{E}_1 - \mathbf{E}_2) = 0,$$

$$\hat{\mathbf{n}} \cdot (\mathbf{D}_1 - \mathbf{D}_2) = \rho_s.$$

Similarly, the boundary conditions for magnetic fields are derived as follows:

$$\hat{\mathbf{n}} \times (\mathbf{H}_1 - \mathbf{H}_2) = \mathbf{J}_s^{imp},$$

$$\hat{\mathbf{n}} \cdot (\mathbf{B}_1 - \mathbf{B}_2) = 0,$$

where $\hat{\mathbf{n}}$ is a unit normal vector from medium 1 to medium 2; ρ_s is the imposed surface charge density; $\mathbf{J}_s^{imp}$ is the imposed surface electric current density.

The main types of possible boundary conditions are as follows.

- *Perfect Electric Conductor (PEC).* This boundary condition models an interface with a perfectly conducting region. Since a perfect conductor cannot hold any electric fields in it, the tangential component of the electric field on the boundary must be zero:

$$\hat{\mathbf{n}} \times \mathbf{E} = 0 \quad \text{on} \quad \Gamma_D. \tag{2.9}$$

- *Impressed surface current.* An antenna can be modeled by prescribing an impressed current on the surface boundary of the antenna, and expressed by

$$\hat{\mathbf{n}} \times \mathbf{H} = \mathbf{J}^{\text{imp}} \quad \text{on} \quad \Gamma_N. \tag{2.10}$$

Splitting the boundary $\partial\Omega$ into two disjoint boundaries namely Γ_D and Γ_N, which indicate Dirichlet and Neumann boundary conditions, we can complete the variational formula as follows:

$$b(\mathbf{E},\mathbf{F}) = l(\mathbf{F}), \quad \forall \mathbf{F} \in \mathbf{H}(\text{curl},\Omega) \text{ and } \hat{\mathbf{n}} \times \mathbf{F} = 0 \quad \text{on} \quad \Gamma_D, \tag{2.11}$$

where

$$\begin{aligned} \mathbf{E} &\in \mathbf{H}(\text{curl},\Omega) \text{ and } \hat{\mathbf{n}} \times \mathbf{E} = \hat{\mathbf{n}} \times \mathbf{E}_D \quad \text{on} \quad \Gamma_D, \\ b(\mathbf{E},\mathbf{F}) &= \int_\Omega \left(\frac{1}{\mu_r} \nabla \times \mathbf{E} \cdot \nabla \times \mathbf{F} - k_0^2 (\varepsilon_r - j\varepsilon_\sigma) \mathbf{E} \cdot \mathbf{F} \right) d\mathbf{x}, \\ l(\mathbf{F}) &= -jk_0 Z_0 \int_\Omega \mathbf{J}^{\text{imp}} \cdot \mathbf{F} d\mathbf{x} + jk_0 Z_0 \int_{\Gamma_N} \mathbf{J}_S^{\text{imp}} \cdot \mathbf{F} dS. \end{aligned} \tag{2.12}$$

2.1.3 Non-dimensionalization

We introduce the following non-dimensional quantities:

$$\mathbf{x} := \frac{\mathbf{x}}{a}, \quad \omega := k_0 a, \quad \mathbf{E} := \frac{\mathbf{E}}{E_0}, \quad \mathbf{J}^{\text{imp}} := \frac{aZ_0}{E_0} \mathbf{J}, \quad \mathbf{J}_s^{\text{imp}} := \frac{Z_0}{E_0} \mathbf{J}_s, \tag{2.13}$$

where a is a characteristic length, and E_0 is a characteristic electric field intensity. For our problem of interest, the scales are naturally determined by the size of the human head and the intensity of incident electric field representing the antenna of a cell phone. Note that the non-dimensional angular frequency coincides with the non-dimensional free space wave number.

$a = 1m$	$\omega = 18.8624$ (0.9 GHz)		$\omega = 37.7248$ (1.8 GHz)		$\omega = 104.7911$ (5.0 GHz)	
	ε_r	ε_σ	ε_r	ε_σ	ε_r	ε_σ
Blood	61.360718	30.718839	59.372261	20.409070	53.950397	19.397026
Skin (Dry)	41.405334	17.311626	38.871857	11.831546	35.773590	11.004411
CSF	68.638336	48.185794	67.200493	29.196078	61.952278	23.716978
Brain	45.805496	15.308879	43.544899	11.515098	39.295105	12.508021
Skull	16.620754	4.826096	15.561986	4.311201	13.045326	4.985904
Muscle	55.955475	19.356274	54.442284	13.874880	50.132092	15.245332

Table 2.1: Non-dimensionalized dielectric parameters of biological tissues [1].

The variational formula, which is rewritten with the non-dimensional variables, retains its original form given by:

$$\begin{aligned} &\mathbf{E} \in \mathbf{H}(\text{curl}, \Omega) \text{ and } \hat{\mathbf{n}} \times \mathbf{E} = \hat{\mathbf{n}} \times \mathbf{E}_D \quad \text{on} \quad \Gamma_D, \\ b(\mathbf{E}, \mathbf{F}) &= \int_\Omega \left(\frac{1}{\mu_r} \nabla \times \mathbf{E} \cdot \nabla \times \mathbf{F} - \omega^2 (\varepsilon_r - j\varepsilon_\sigma) \mathbf{E} \cdot \mathbf{F} \right) d\mathbf{x}, \\ l(\mathbf{F}) &= -j\omega \int_\Omega \mathbf{J}^{\text{imp}} \cdot \mathbf{F} d\mathbf{x} + j\omega \int_{\Gamma_N} \mathbf{J}_S^{\text{imp}} \cdot \mathbf{F} dS. \end{aligned} \tag{2.14}$$

Table 2.1 shows the range of the non-dimensional material constants relevant for the problem of interest.

2.2 Perfectly Matched Layer

For scattering problems, Maxwell equations are formulated in the unbounded domain, and the scattered field is required to satisfy the Silver-Müller radiation condition at infinity:

$$\lim_{r \to \infty} r[\nabla \times \mathbf{E} - jk_0 \mathbf{e}_r \times (\mathbf{e}_r \times \mathbf{E})] = 0,$$

where r is a spherical coordinate centered at the scatterer. This radiation condition represents the requirement that the wave should propagate outward; the radiating

electric and magnetic fields are apt to be transverse in the far-field region, which also implies that the Poynting vector becomes radial. In addition to Maxwell equations, this radiation condition should be imposed to ensure that the scattered field is uniquely determined.

However, the Finite Element (FE) discretization can be applied only in a bounded domain. Consequently, the infinite domain has to be truncated via an artificial boundary layers mimicking the radiating conditions. A number of techniques have been invented over the last three decades including coupling with Boundary Elements, Infinite Elements [18], and Absorbing Boundary Conditions [32]. In this work, we shall use perhaps the most popular and effective Perfectly Matched Layer (PML) [12] technique and follow the construction presented by Matuszyk and Demkowicz [60].

PML formulation in spherical coordinates. Suppose our domain of interest $\Omega \subset \mathbb{R}^3$ is truncated with a sphere of radius a, and the PML complex stretching is applied to the exterior domain Ω_{PML} where $r > a$ as depicted in Fig. 2.1. The stretching will convert physically correct outgoing waves into evanescent waves decreasing exponentially away from the scatterer, with a simultaneous exponential amplification of the incoming waves. A consecutive application of the PEC condition at $r = b > a$ eliminates the exponentially blown-up of incoming waves.

First, we define the complex variable, $z(r)$ through complex stretching of

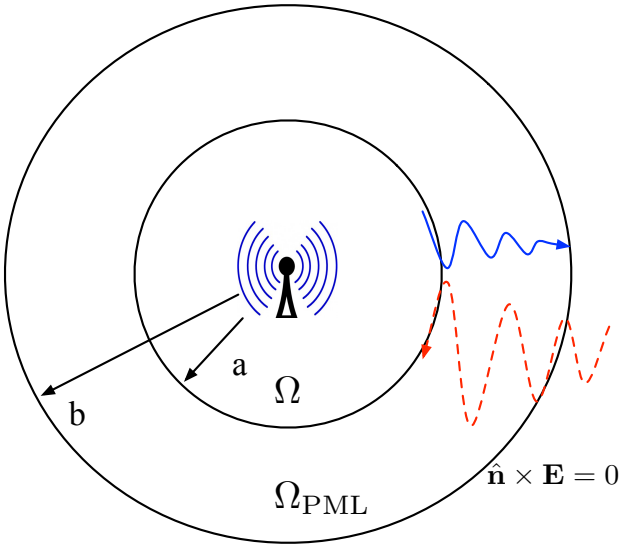


Figure 2.1: Illustration of the Perfectly Matched Layer (PML).

the real variable r, described as follows:

$$z(r) = \begin{cases} r & 0 \leq r < a \quad \text{domain of interest,} \\ \left(1.0 - \frac{j}{k}\left(\frac{r-a}{b-a}\right)^{\alpha}\right) r & a \leq r \leq b \quad \text{PML layer,} \end{cases}$$

where α represents a tuning parameter. The complex coordinate stretching is constructed in such a way that the outgoing wave reaches a zero (to machine-precision) at $r = b$ so that no wave is reflected from the exterior boundary. This leads to a stretching map $\mathbf{x}_C : \Omega \ni \mathbf{x} \rightarrow \tilde{\mathbf{x}} \in \tilde{\Omega}$ from real coordinates $\mathbf{x}$ to the complex-stretched coordinates $\tilde{\mathbf{x}}$. Analogus to the Piola mapping between a master element and a corresponding physical element, the complex-stretched mapping can be applied in the same way.

Next, we compute the Jacobian corresponding to the complex-stretching. Since we compute in Cartesian coordinates and stretch them in spherical coordi-

nates, the following chain rules are used to determine the complex variable Jacobian:

$$\tilde{\mathbf{J}} = \frac{\partial \tilde{\mathbf{x}}}{\partial \mathbf{x}} = \frac{\partial \tilde{\mathbf{x}}}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial \mathbf{r}} \cdot \frac{\partial \mathbf{r}}{\partial \mathbf{x}},$$

where $\mathbf{x}$ and $\tilde{\mathbf{x}}$ are defined as follows

$$\begin{cases} x &= r\sin\theta\cos\phi \\ y &= r\sin\theta\sin\phi \\ z &= r\cos\theta \end{cases}, \quad \begin{cases} \tilde{x} &= z(r)\sin\theta\cos\phi \\ \tilde{y} &= z(r)\sin\theta\sin\phi \\ \tilde{z} &= z(r)\cos\theta. \end{cases}$$

Recalling the Piola maps transforming master elements onto physical elements, the same exact sequence logic should be applied to the stretched transformation:

$$\tilde{\mathbf{E}} \circ \mathbf{x}_C = \tilde{\mathbf{J}}^{-T}\mathbf{E} \text{ and } (\tilde{\nabla} \times \tilde{\mathbf{E}}) \circ \mathbf{x}_C = \tilde{J}^{-1}\tilde{\mathbf{J}}(\tilde{\nabla} \times \mathbf{E}). \tag{2.15}$$

Since the electric field $\mathbf{E}$ in the PML region satisfies the stretched version of the Maxwell equations, we obtain the following modified bilinear form:

$$\begin{aligned} \tilde{b}\left(\tilde{\mathbf{E}}, \tilde{\mathbf{F}}\right) &= \int_{\Omega} \left(\frac{1}{\mu_r} \tilde{\nabla} \times \tilde{\mathbf{E}} \cdot \tilde{\nabla} \times \tilde{\mathbf{F}} - \omega^2 (\varepsilon_r - j\varepsilon_\sigma) \tilde{\mathbf{E}} \cdot \tilde{\mathbf{F}} \right) \tilde{J} d\mathbf{x} \\ &= \int_{\Omega} \left(\frac{1}{\mu_r} (\tilde{J}^{-2} \tilde{\mathbf{J}}^T \tilde{\mathbf{J}}) \nabla \times \mathbf{E} \cdot \nabla \times \mathbf{F} - \omega^2 (\varepsilon_r - j\varepsilon_\sigma) (\tilde{\mathbf{J}}^{-1} \tilde{\mathbf{J}}^{-T}) \mathbf{E} \cdot \mathbf{F} \right) \tilde{J} d\mathbf{x} \\ &= \int_{\Omega} \left(\frac{1}{\mu_r} (\underbrace{\tilde{J}^{-1} \tilde{\mathbf{J}}^T \tilde{\mathbf{J}}}_{\bar{A}^{-1}}) \nabla \times \mathbf{E} \cdot \nabla \times \mathbf{F} - \omega^2 (\varepsilon_r - j\varepsilon_\sigma) (\underbrace{\tilde{J} \tilde{\mathbf{J}}^{-1} \tilde{\mathbf{J}}^{-T}}_{\bar{A}}) \mathbf{E} \cdot \mathbf{F} \right) d\mathbf{x}. \end{aligned}$$

Introducing a metric tensor $\bar{A} = \tilde{J}\tilde{\mathbf{J}}^{-1}\tilde{\mathbf{J}}^{-T}$, the above bilinear form can be rewritten as follows:

$$\tilde{b}\left(\tilde{\mathbf{E}}, \tilde{\mathbf{F}}\right) = \int_{\Omega} \left(\frac{1}{\mu_r} \bar{A}^{-1} \nabla \times \mathbf{E} \cdot \nabla \times \mathbf{F} - \omega^2 (\varepsilon_r - j\varepsilon_\sigma) \bar{A} \mathbf{E} \cdot \mathbf{F} \right) d\mathbf{x}. \tag{2.16}$$

This implies that the complex-stretching changes the transmitting medium in which the original Maxwell systems is posed into an anisotropic one.

2.3 BioHeat Transfer

The temperature rise in biological tissues exposed to Electro-Magnetic (EM) waves can be determined by solving the BHT equation. The BHT model is characteristic in two unique mechanisms: blood perfusion and metabolism. Additionally, the model is also dependent on external heating sources such as EM energy deposition. In this section, we consider the Pennes BHT model accounting for blood perfusion, metabolism, and Specific Absorption Rate (SAR) induced by EM waves.

2.3.1 The Pennes model

The Pennes BHT model [72] assumes that the net heat transfer $\dot{q}_{\text{perf}}$ between the blood and the tissue in a control volume Ω_{body} is proportional to the temperature difference between the arterial blood entering the volume and the venous blood leaving the tissue. The constant of proportionality includes a scalar parameter W_b, defined as the volumetric rate of blood perfusion. In this sense, the blood is modeled as being locally distributed as scalar sources or sinks of heat,

$$\dot{q}_{\text{perf}} = W_b c_b (u_a - u_v), \tag{2.17}$$

where

$$\begin{aligned}
\dot{q}_{\text{perf}} &= \text{net heat transfer in a control volume} && [W/m^3],\\
W_b &= \text{volumetric blood perfusion rate} && [kg/m^3 s],\\
c_b &= \text{specific heat of blood} && [J/kg\,^{\circ}\text{C}],\\
(u_a - u_v) &= \text{temperature difference between the arteries and veins} && [^{\circ}\text{C}].
\end{aligned}$$

In addition to the standard Pennes model, deposited energy by EM waves transmitted in the tissue is included as SAR:

$$\dot{q}_{\mathrm{SAR}} = \rho \mathrm{SAR} = \frac{\sigma |\mathbf{E}|^2}{2}, \tag{2.18}$$

where

$$\begin{aligned} \sigma &= \text{conductivity of the tissue} && [\Omega/m], \\ \mathbf{E} &= \text{electric field intensity} && [V/m], \\ \rho &= \text{density of the tissue} && [kg/m^3]. \end{aligned}$$

Then, the energy conservation law in a control volume of the tissue yields the BHT equation that accounts for the volumetric blood flow, metabolism, and localized heating by EM waves:

$$\rho c \frac{\partial u}{\partial t} = \underbrace{\nabla \cdot k \nabla u}_{\text{diffusion of heat}} + \underbrace{W_b c_b (u_{a0} - u)}_{\text{perfusion rate}} + \underbrace{\dot{q}_m}_{\text{metabolism}} + \underbrace{\dot{q}_{\mathrm{SAR}}}_{\text{EM energy}}, \tag{2.19}$$

where

$$\begin{aligned} k &= \text{effective thermal conductivity of the tissue} && [W/m^\circ\mathrm{C}], \\ c &= \text{specific heat of the tissue} && [J/kg^\circ\mathrm{C}], \\ u &= \text{temperature of veins in the tissue} && [^\circ\mathrm{C}]. \end{aligned}$$

In general, the thermal properties of the tissue and blood are experimentally determined. We also assume that the arterial temperature u_{a0} is constant in the tissue, and metabolic heat generation is given.

2.3.2 Variational formulation

For the simplicity, the following dimensionless variables are defined with a reference time t_0, a reference length a and a reference temperature u_0:

$$\alpha := \frac{kt_0}{\rho c a^2}, \quad \beta := \frac{W_b c_b t_0}{\rho c}, \quad \theta := \frac{u - u_{a0}}{u_0}, \quad h := \frac{ht_0}{\rho c a}, \quad t := \frac{t}{t_0}, \quad \mathbf{x} := \frac{\mathbf{x}}{a}. \tag{2.20}$$

Then, the Pennes equation in (2.19) can be rewritten:

$$\dot{\theta} = \nabla \cdot (\alpha \nabla \theta) - \beta \theta + \dot{q}, \text{ and } \dot{q} := \frac{(\dot{q}_m + \dot{q}_{\mathrm{SAR}})\, t_0}{\rho c u_0}. \tag{2.21}$$

Multiplying (2.21) by a space test function $\psi \in H^1$, and integrating by parts over the domain of interest, Ω_{body}, we obtain

$$\int_{\Omega} \left(\dot{\theta}\psi + \alpha \nabla \theta \cdot \nabla \psi + \beta \theta \psi \right) d\mathbf{x} + \int_{\partial\Omega} \alpha \nabla \theta \cdot \psi_n dS = \int_{\Omega} \dot{q} \psi d\mathbf{x}. \tag{2.22}$$

The boundary condition should be specified accounting for the heat exchange with external environments. The convective boundary condition with a prescribed external temperature θ_{air} is specified as a Cauchy boundary condition:

$$\mathbf{n} \cdot (\alpha \nabla \theta) = h\,(\theta - \theta_{air})\,, \tag{2.23}$$

where h represents the heat transfer coefficient on the surface of the domain.

Integrating over time, we obtain the following variational form accounting for the boundary Γ_D and Γ_C, which denote Dirichlet and Cauchy boundaries:

$$\left(\dot{\theta}, \psi\right) + b\,(\theta, \psi) = l\,(\psi)\,, \quad \forall \psi \in H^1(\Omega) \text{ and } \psi = 0 \quad \text{on} \quad \Gamma_D, \tag{2.24}$$

where $(\cdot,\cdot)$ denotes the L^2-inner product on Ω and

$$\begin{aligned} &\theta \in H^1(\Omega), \quad \dot{\theta} \in L^2(\Omega) \text{ and } \theta = \theta_D \quad \text{on} \quad \Gamma_D, \\ b(\theta,\psi) &= \int_\Omega (\alpha \nabla\theta \cdot \nabla\psi + \beta\theta\psi)\, d\mathbf{x} + \int_{\Gamma_C} h(\theta - \theta_{\text{air}})\,\psi dS, \\ l(\psi) &= \int_\Omega \dot{q}\psi d\mathbf{x}. \end{aligned} \tag{2.25}$$

2.3.3 Coupling through material properties

Attempting to model the complex BHT involving EM waves in living tissue is challenging due to the interdependent nature of the material properties. More precisely,

- in the EM problem, the temperature is dependent on the local SAR distribution;
- the SAR is dependent on local EM properties, which are also a function of the local tissue temperature, water content, and other factors.

In the present study, coupling is made in two steps: first, the SAR distribution in the human head is computed; next, the temperature distribution is determined taking into account of blood perfusion, metabolism, and EM energy deposited in the tissue.

The temperature dependence of the tissue thermal properties is not considered because the temperature range increased by mobile devices is expected to be sufficiently small (below a °C degree). A list of thermal properties of the human tissues is tabulated in Table 2.2.

	$\rho[kg/m^3]$	$c[J/kg\,^\circ\mathrm{C}]$	$k[W/m\,^\circ\mathrm{C}]$	$\dot{q}_m[W/m^3]$	$W_b c_b[W/m^3\,^\circ\mathrm{C}]$
Air	1.16	1006	0.0263	0	0
Brain	1039	3700	0.57	7100	40000
White matter	1043	3600	0.5	7100	40000
Fat	916	2500	0.25	300	1700
Muscles	1041	3600	0.5	480	2700
Skin	1100	3500	0.42	1620	9100
Skull	1645	1300	0.4	590	3300

Table 2.2: Material thermal properties of various biological tissues [43].

However, the temperature dependence of the dielectric properties and conductivity in the tissue can be observed during the time integration. This weak coupling mechanism is justified by noting the different time scale between the EM field and transient temperature field in BHT. While the EM field reaches a steady-state in micro seconds, the temperature field is transient over a range of seconds or minutes.

Chapter 3

3D hp-Finite Element Technologies for Hybrid Meshes

The hp-adaptive Finite Element Method (FEM), which controls locally both the element size h and the polynomial order p, can lead to substantial savings in the cost of both computation and storage, at the expense of increasing complexity of algorithms and implementations. A few hp-adaptive codes have been developed and have demonstrated the efficiency of the overall idea. However, technical problems such as precise geometry description, generation of curved meshes, and dynamic hp-mesh refinements still remain as algorithmic and software engineering challenges. Particularly, three-dimensional mesh refinements for unstructured hybrid meshes are rarely implemented in practice due to these barriers.

While most academic research projects for the adaptive methods focus on delivering precise error estimates to achieve a fast convergence, little attention has been paid to

- whether such refinements can be applied to target elements satisfying necessary mesh regularity assumptions,
- whether these processes can be repeated through the entire adaptive procedure.

The answer is not straightforward when we deal with unstructured hybrid meshes that include the elements of all shapes (*i.e.*, hexahedra, tetrahedra, prisms and pyramids). A major challenge in implementing a general refinement scheme lies in how to keep track of dynamically changing mesh connectivities. In particular, anisotropic refinements may create an incompatible refinement pattern so that the adaptive procedure cannot proceed further, a situation known as a *deadlock*.

In this chapter, we present general mesh refinement algorithms that support both isotropic and anisotropic mesh refinements on three-dimensional unstructured hybrid *hp*-meshes, and discuss their implementation. To the best of our knowledge, our 3D *hp*-code is the only code that supports the both isotropic and anisotropic refinements on a hybrid mesh that includes elements of all shapes.

3.1 Code overview: `hp3d`

The `hp3d` code, a general software package for the *hp*-adaptive FEM, has been under development for the last three years by the author in collaboration with P. Gatto and L. Demkowicz. To facilitate the adaptive workflow described in Fig. 3.1, a basic modular framework for the refinement package is designed as follows:

Geometry. An initial mesh triangulation is generated once from a given input geometry. Unlike for the standard FE procedure, the geometry information is continuously updated as a sequence of *hp*-meshes is generated through local refinements. To facilitate an interactive relationship between *hp*-mesh and

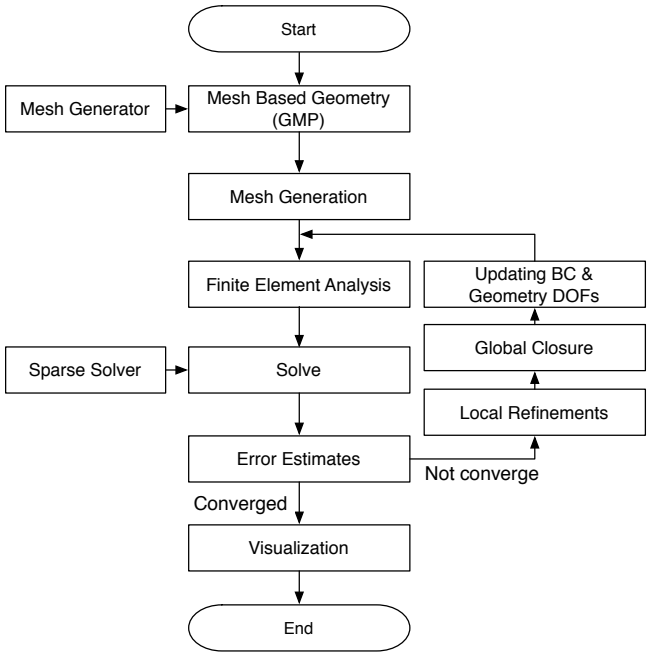


Figure 3.1: A general workflow in *hp*-adaptive FEM.

the reference geometry, the Geometry Modeling Package (GMP) [99] provides a Mesh-Based Geometry (MBG) [70] interface: the geometry is treated as a collection of blocks that topologically form a mesh-like structure, with globally C^0-continuous parameterizations provided in each block.

Data structure. The data structure contains only two array structures: initial mesh elements and nodes [1]. The `NODES` array supports information on nodal trees originated from mesh refinements. While the initial mesh elements correspond to the MBG description in GMP, nodal trees store the hierarchical relations for a series of meshes updated by *h*-refinements. Each node in-

[1] An abstraction for all topological entities: vertices, edges, faces and element interiors.

cludes an independent polynomial order to support variable order elements. Extending the previous work for hexahedral meshes [23, 28], the current data structure is a generalization designed to support the elements of all shapes along with both isotropic and anisotropic refinements.

Mesh modification. In providing the various mesh refinements for unstructured meshes, complexity is the major obstacle. This module contains generic interfaces and lookup tables to perform diverse kinds of isotropic and anisotropic refinements for elements of all shapes. Refinement algorithms are designed to support one irregular meshes, and the code provides the constrained approximation technique [75] to resolve one-level hanging nodes.

In the following sections, we provide detailed descriptions for the data structure and mesh modification modules.

3.2 Data abstraction for unstructured hybrid *hp*-meshes

The data structure in `hp3d` is designed to represent a general, unstructured 3D hybrid mesh consisting of the elements of all shapes *i.e.*, the hexahedron, prism, pyramid, and tetrahedron.

3.2.1 Mesh topology

A conventional mesh data structure, which consists of vertex nodes and element-to-nodes connectivities, is not appropriate for the *hp*-FEM. In the adaptive context, an *hp*-mesh evolves via local mesh refinements; element-to-nodes con-

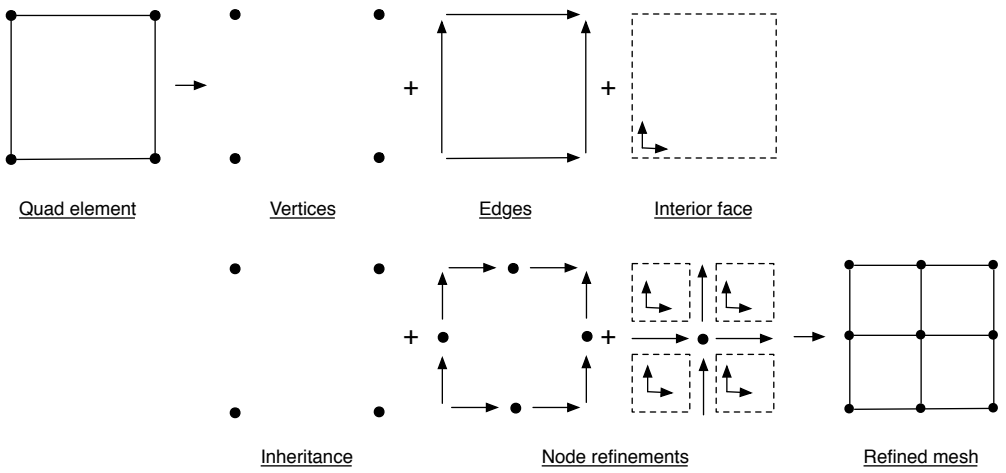


Figure 3.2: Topological representation of a quadrilateral element and its refinement procedure.

nectivities are reconstructed from dynamically updated data structure array `NODES`. To enable the reconstruction of connectivities, the concept of an (abstract) node is extended to include all topological entities: vertices, edges, faces and elements. Specifically, an element is defined as a set of topological nodes enclosing the interior of the element. Namely,

$$\text{Element} = \text{Vertices} + \text{Edges} + \text{Faces} + \text{Interior node}.$$

In this topological framework, mesh refinements translate into a set of orchestrated node refinements. As seen in Fig. 3.2, a quadrilateral element is first decomposed into associated topological nodes; next, nodes are individually refined appropriately.

In addition, storing all topological entities is advantageous in supporting different energy spaces and discretizations arising from multi-physics problems.

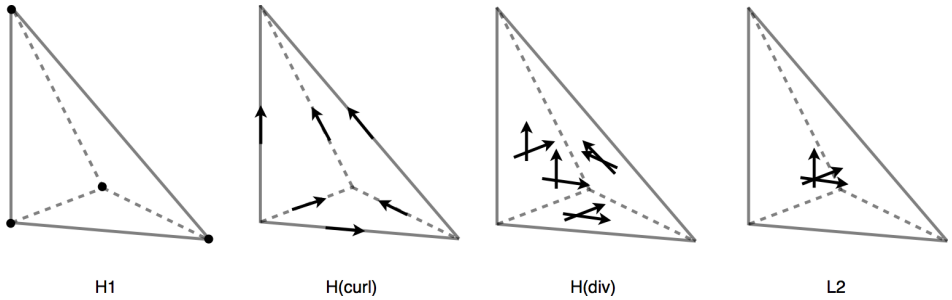


Figure 3.3: The lowest DOFs associated with different topological entities according to the selected energy space and discretization.

Figure 3.3 illustrates the lowest order elements corresponding to the well-known exact sequence:

$$H^1 \xrightarrow{\nabla} \mathbf{H}(\text{curl}) \xrightarrow{\nabla\times} \mathbf{H}(\text{div}) \xrightarrow{\nabla\cdot} L^2.$$

For instance, the lowest order H^1-conforming elements have Degrees of Freedom (DOFs) at vertex nodes and provide a continuous scalar-valued space; $\mathbf{H}(\text{curl})$-conforming elements (known as Nédélec elements) employ edge-based vector-valued basis functions. Similarly, DOFs for $\mathbf{H}(\text{div})$-conforming and L^2 elements are assigned to face nodes and an interior node respectively. This characteristic feature of the exact sequence elements naturally leads to the concept of the abstract node including all topological entities. The element shape functions and corresponding DOFs are then grouped into subsets corresponding to its nodes: vertices, edge, face and element interiors. This results in a natural definition of the object called "node". For details on higher order FEs and the exact sequence properties, we refer to [27, 80].

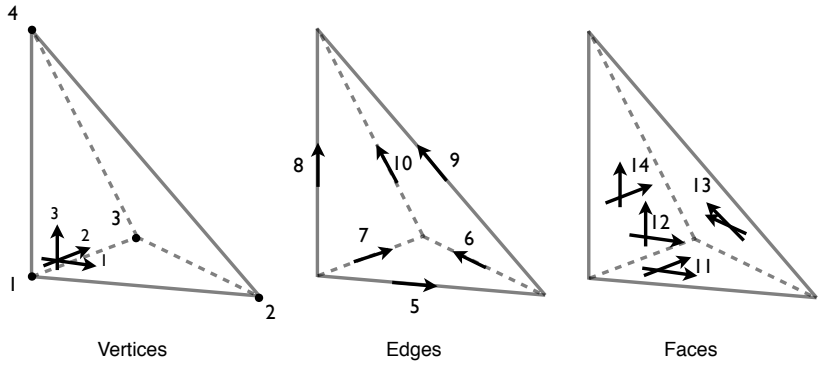
Adjacency(in::type, in::node)		
-	EDGE_TO_VERT	FACE_TO_VERT
VERT_TO_EDGE	-	FACE_TO_EDGE
VERT_TO_FACE	EDGE_TO_FACE	-

Table 3.1: Adjacency relations of the element-to-node connectivity.

3.2.2 Definition of an element: an object oriented approach

To facilitate hybrid meshes, elements of particular shapes are defined in an abstract way in terms of their topologies and (master element) geometry. A complete set of adjacency relations is provided for each element type as tabulated in Table 3.1. An example for a tetrahedral element is shown in Fig. 3.4. The edge-to-vertex and face-to-vertex connectivities define also *local*, master element orientations of element edges and faces [2]. Globally, each edge, face and element in the physical domain comes with its own system of coordinates. The element coordinates coincide topologically with the corresponding master element coordinates. Given an element, however, the corresponding element local parameterizations of its edges and faces (as implied by the element coordinates) will differ in general from the (global) edge and face coordinates. One of the key constructions needed in FE computations is the local-to-global change of coordinates for element edges and faces. These correspond to what we call the *node (edge and face) orientations*. As an example, Fig. 3.5 defines six different orientations for a triangular node with the corresponding re-enumeration of vertex nodes. The triangle vertices determine its local coordinates while the global coordinates are indicated with axes 1,2.

[2] In other words, the master element edge and face parameterizations.



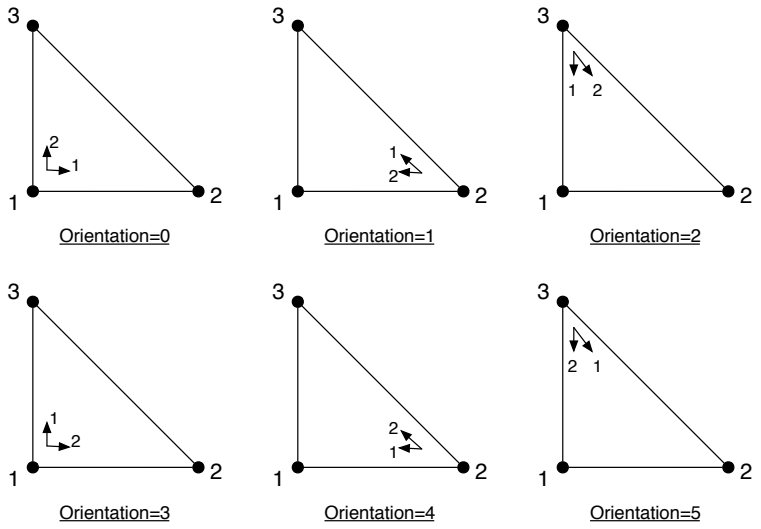
Local (master) enumeration of nodes

"tetr"	in	out	"tetr"	in	out
EDGE_TO_VERT	6	[2,3]	VERT_TO_EDGE	2	[5,6,9]
FACE_TO_VERT	13	[2,3,4]	VERT_TO_FACE	2	[11,12,13]
FACE_TO_EDGE	13	[6,10,9]	EDGE_TO_FACE	6	[11,13]

Figure 3.4: Adjacency relations on a tetrahedral element.

3.2.3 Nodal trees

Mesh refinements translate into creation of new nodes and growth of corresponding nodal trees [28, 76]. For example, Fig. 3.6 depicts a series of 2D *h*-refinements along with the corresponding growth of nodal trees. Unlike other adaptive codes [50, 71], we do not rely on global lookup tables (*e.g.*, hash-tables and binary-search trees) to keep track of the dynamically evolving connectivities. Instead, the connectivities are reconstructed from those nodal trees. Detailed algorithms will be discussed in Section 3.4.



Orientation	0	1	2	3	4	5
Local2Global	[1,2,3]	[2,3,1]	[3,1,2]	[1,3,2]	[2,1,3]	[3,2,1]

Figure 3.5: Re-enumeration of vertex nodes accounting for the global orientations.

3.3 Refinement patterns

A refinement pattern defines node hierarchies and orientations of child nodes, and their enumerations. As an element is seen as a collection of nodes, the mesh refinement procedure can be described breaking a set of nodes. Naturally, refinement patterns are incrementally constructed to support higher dimensional node shapes. Various refinements for 2D faces are illustrated in Fig. 3.7. Note that when a *parent node* is split into its *child nodes*, the child nodes are setup with orientations consistent with that of the parent node, the so called a *inheritance rule*. While the inheritance rule provides a natural (implicit) definition of child nodes orientations

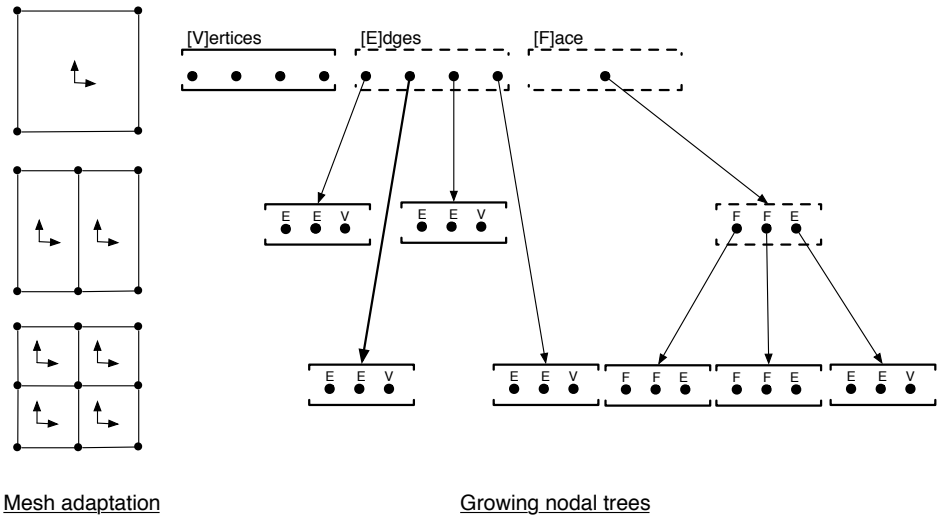


Figure 3.6: Representation of *h*-refinements using a family of nodal trees.

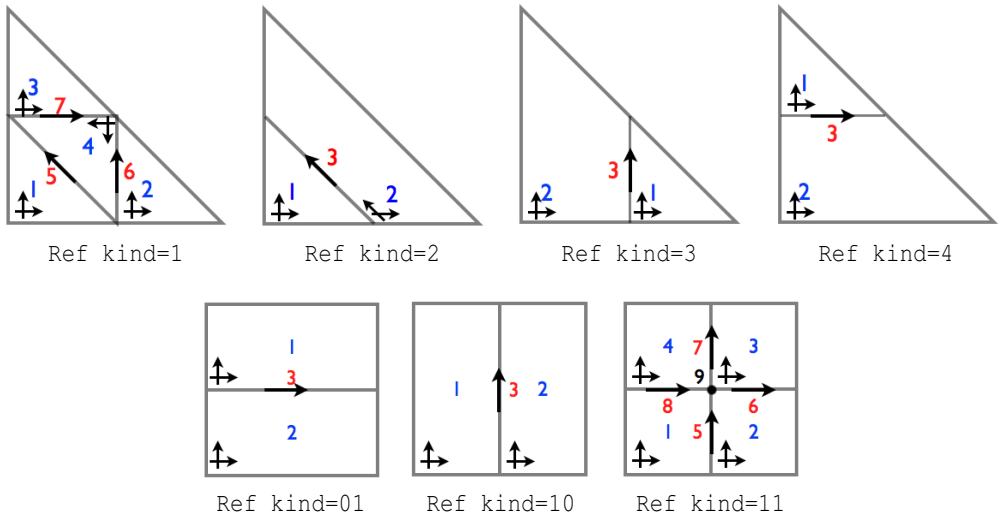


Figure 3.7: Supported 2D refinement patterns.

for the illustrated cases, we have encountered situations [3] where there is no natu-

[3]For instance, when defining three isotropic refinements of a parent tetrahedron into eight child tetras depicted in Fig. 3.9.

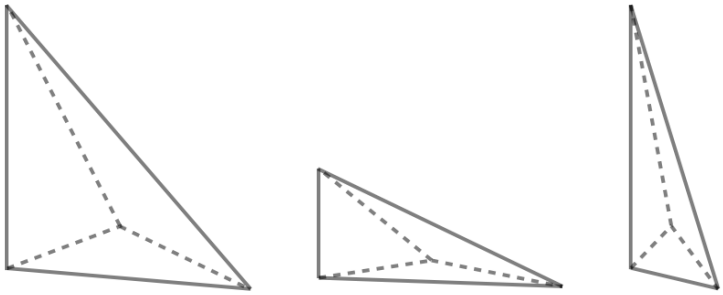


Figure 3.8: Various shapes of tetrahedral elements.

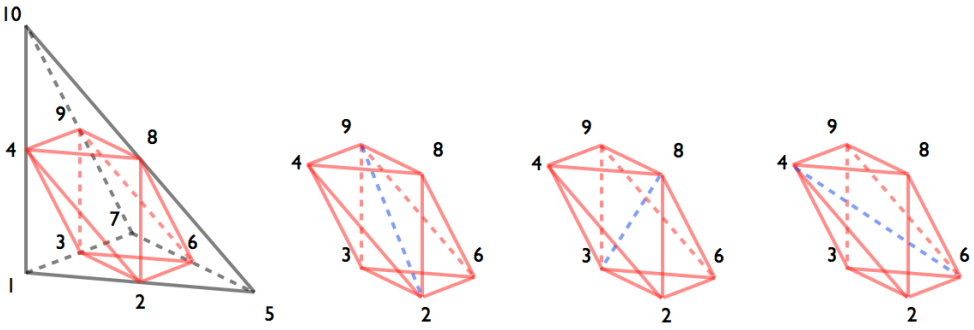
rally defined inheritance rule. Consequently, we had to introduce appropriate look up tables that define the child nodes orientations explicitly.

Tetrahedron refinements.

Complex 3D geometries are most commonly represented with unstructured tetrahedral meshes for which a number of supporting algorithms has been developed [7, 56, 84, 85]. For a detailed review of those methods, we refer [69].

Unstructured tetrahedral meshes may contain badly shaped tetrahedral objects (*e.g.*, a wedge or a spire) as depicted in Fig. 3.8. It is well-known that such elements affect the convergence and accuracy of discrete solutions [8, 86]. Thus, a refinement package should be able to control the shape regularity so that the refined mesh is not degenerate.

Unlike for hexahedral or prismatic meshes, tetrahedral child elements need not inherit the shape regularity from their parent, even in the case of an isotropic refinement. Several research projects [51, 73] were devoted to the development of



Tetr.	Ref=11	Ref=12	Ref=13	Tria.	Ref=11	Ref=12	Ref=13
1	(1,2,3,4)			9	(2,3,4)		
2	(2,5,6,8)			10	(2,6,8)		
3	(3,6,7,9)			11	(3,6,9)		
4	(4,8,9,10)			12	(4,8,9)		
5	(2,9,4,8)	(3,8,2,6)	(4,6,8,2)	13	(2,9,4)	(3,8,2)	(4,6,2)
6	(2,9,8,6)	(3,8,6,9)	(4,6,2,3)	14	(2,9,8)	(3,8,6)	(4,6,3)
7	(2,9,6,3)	(3,8,9,4)	(4,6,3,9)	15	(2,9,6)	(3,8,9)	(4,6,9)
8	(2,9,3,4)	(3,8,4,2)	(4,6,9,8)	16	(2,9,3)	(3,8,4)	(4,6,8)

Edge.	Ref=11	Ref=12	Ref=13		
17	(2,9)	(3,8)	(4,6)		

Figure 3.9: Isotropic refinements of tetrahedral elements.

stable refinement schemes for unstructured grids.

Consider the isotropic refinements depicted in Fig. 3.9. First, a tetrahedral element is split into four tetrahedral elements and an octahedron. Next, the octahedron can be refined into four tetrahedral elements in three different ways corresponding to the selected diagonal edge. The first four tetrahedral elements, which are obtained by cutting the four corners, are similar to the parent tetrahedron, and their orientations are also set to be consistent with their parent. In contrast, the tetrahedral elements resulting from the octahedron in the middle are not similar to

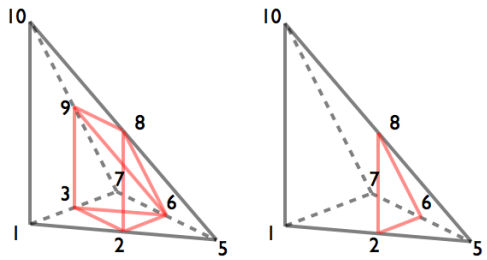


Figure 3.10: Anisotropic refinements of tetrahedral elements. These refinements are specially designed for wedge and spire shapes.

the parent tetrahedron; the elements are not only scaled but also stretched. This also implies that a mesh may degenerate if inappropriate subdivisions are repeated. It has been proved [52, 54] that the shape regularity is preserved if the octahedron is broken by the shortest diagonal edge. This rule has been enforced in our code as well.

For completeness, we also introduce the anisotropic refinements depicted in Fig. 3.10. These anisotropic refinements have been developed to enable refinement of thin membrane structures, see [35].

Prism refinements

A hybrid mesh consisting of tetrahedral and prismatic elements (see, [10, 20, 100]) is often used in modelling boundary layers on a complex geometry. A prismatic layer is created by extruding triangular facets on the surface of 3D objects. This hybridization provides flexibility in managing a complex geometry modeled with tetrahedral elements, providing an opportunity to achieve cost-effective so-

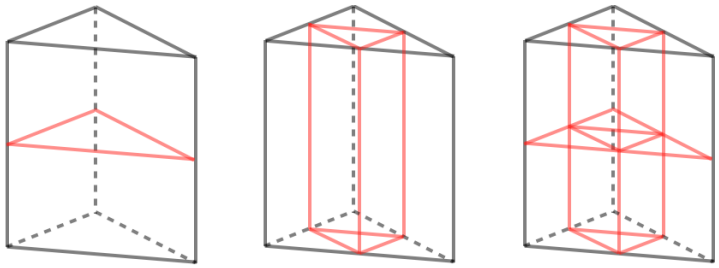


Figure 3.11: Isotropic and anisotropic refinements of prismatic elements.

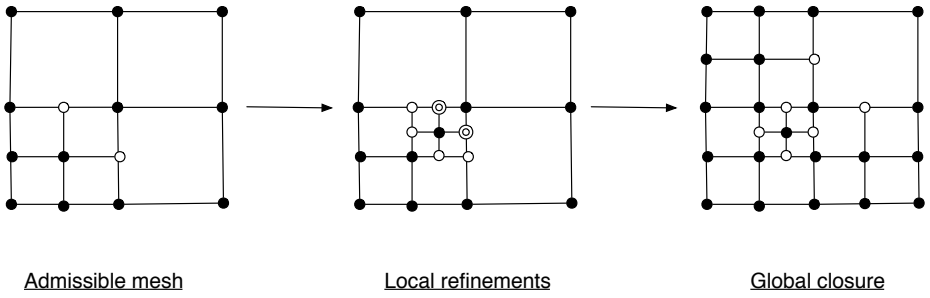


Figure 3.12: Two-step refinement procedure.

lutions by using anisotropic *hp*-refinements on prismatic elements as depicted in Fig. 3.11.

We skip the discussion on similar refinements that have been developed for hexahedra and pyramids [35], since these are not used in this study.

3.4 Algorithm description

In this section, we describe refinement algorithms. Our refinement scheme allows one-level hanging nodes and completes within two steps, see Fig. 3.12:

1. *Local refinements.* Selected elements are refined. As a result, the refined mesh may include arbitrary level hanging nodes.

2. *Global mesh closure.* By enforcing additional refinements, the refined mesh recovers one-level mesh irregularity.

In actual FE computations, the one-level hanging nodes are appropriately resolved by enforcing constraints providing the continuity of the solution across the inter-element boundaries [75]. As no DOF is assigned to hanging nodes, we also identify them as *inactive* nodes. Solution DOFs are then constrained (or interpolated) by *active* DOFs on their parent and neighboring nodes. In the following sections we describe the algorithms involved in more detail.

3.4.1 Reconstruction of nodal connectivities

As mentioned earlier, we do not store element-to-nodes connectivities that are dynamically modified during the adaptive refinements. Instead, we provide a reconstruction scheme to retrieve updated connectivities. The basic idea is to consider the refined mesh as a locally structured mesh where the connectivities within that local structure is ruled by a refinement logic. Given an element, the element-to-nodes connectivities for the element and its ancestors are partially (locally) reconstructed by traversing over a family of nodal trees. Traversing upward in the nodal trees, we reach the initial mesh ancestor for which the connectivities are stored explicitly. We back traverse the refinement trees reproducing virtually the stored node refinements and reconstructing the connectivities.

```
!> The routine create new nodes and associate them as child nodes
!! under the input nod.
subroutine break_node(in::nod, in::refinement)
  !! nod, refinement - target node and refinement flag

  do i=1, refinement.number_of_children
    child = mesh.insert_node(refinement.child_type(i))
    nod.associate(child)
  end do

  !! record the applied refinement in the input node
  nod.set_refinement(refinement)
end subroutine elem_nodes
```

Figure 3.13: Node breaking procedure.

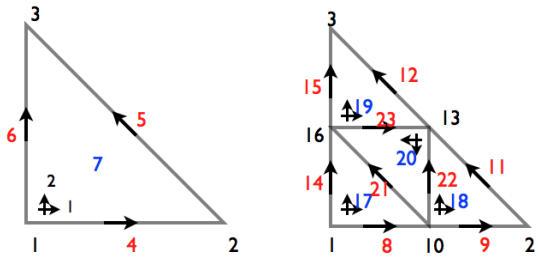


Figure 3.14: Node enumeration in the isotropic refinement of a triangular face node.

Before explaining the detailed reconstruction scheme, we first briefly describe the element/node refinement procedure. When a node is refined, new nodes are created and they are hierarchically associated with the node as its children. We give pseudo code for this in Fig. 3.13. An important observation is that the refined element connectivities can be explained by applied refinements. In practice, the connectivity logic for each refinement pattern is encoded into three lookup arrays; namely `PARENT`, `CHILD`, and `ORIENTATION` [4]. For a 2D example, consider an isotropic refinement applied to a triangular element depicted in Fig. 3.14. Starting

[4]The lookup arrays are mechanically generated by exhaustive search algorithms.

with a set of lookup arrays, we can reconstruct the element-to-node connectivity of the first triangular child node (`node number 17`) as follows.

```
Encoded lookup arrays for the 1st child triangular node:
  PARENT     (1:7)= { 1,4,6,  4,7,6,  7 }
  CHILD      (1:7)= { 0,3,3,  1,5,1,  1 }
  ORIENTATION(1:7)= { 0,0,0,  0,0,0,  0 }

Reconstruction rule at i-th node:
  nodes(i) = mesh.nodes(PARENT(i)).child(CHILD(i))

Reconstruction of the first child node:
  nodes(1) = mesh.nodes(PARENT(1)=1).child(CHILD(1)=0) =  1
  nodes(2) = mesh.nodes(PARENT(2)=4).child(CHILD(2)=3) = 10
  nodes(3) = mesh.nodes(PARENT(3)=6).child(CHILD(3)=3) = 16
  nodes(4) = mesh.nodes(PARENT(4)=4).child(CHILD(4)=1) =  8
  nodes(5) = mesh.nodes(PARENT(4)=7).child(CHILD(5)=5) = 21
  nodes(6) = mesh.nodes(PARENT(5)=6).child(CHILD(6)=1) = 14
  nodes(7) = mesh.nodes(PARENT(7)=7).child(CHILD(7)=1) = 17
```

We stress that this reconstruction scheme is generic for all kinds of refinements and no shape-dependent information is used.

A general procedure for this one-level reconstruction is clearly captured in Fig. 3.15. One can easily realize that the scheme is recursive as described in Fig. 3.16. The scheme traverses nodal trees until it reaches an initial mesh element which stores the full spatial connectivity. After that, the element-to-node connectivity is level-by-level reconstructed going down the trees.

3.4.2 Local refinements

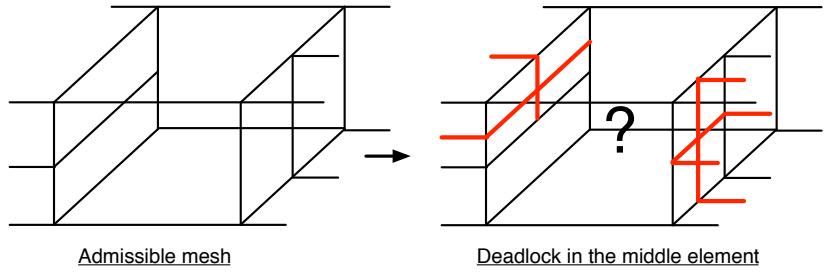
A major challenge in performing 3D hybrid mesh refinements is to avoid a deadlock situation so that mesh refinements can be repeatedly applied. Particularly interesting is the case of anisotropic refinements; a combination of isotropic and

```
!> The subroutine perform one-level reconstruction using nodal trees
!! and refinement lookup tables.
subroutine reconstruct_element2nodes(in::p_nodes, in::p_orients,
                                     in::intr,
                                     out::nodes, out::orients)
  !! p_nodes, p_orients - parent and its nodal connectivity
  !! intr - target element (interior node)
  !! nodes, orients - the reconstructed element to node connectivities

  !! choose lookup tables for given interior node
  call select_lookup(intr, PARENT, CHILD, ORIENTATION)

  !! reconstruct the element-to-node connectivity and orientations
  do i=1,NNODE(intr)
    nod         = mesh.nodes(p_nodes(PARENT(i)))

    !! child node and orientation should be re-enumerated
    !! accounting for the given parent orientation
    nodes(i)    = nod.child(local2global(p_orient(i), CHILD(i)))
    orients(i)  = rotate(p_orient(i), ORIENTATION(i))
  end do
end subroutine reconstruct_elem_nodes
```

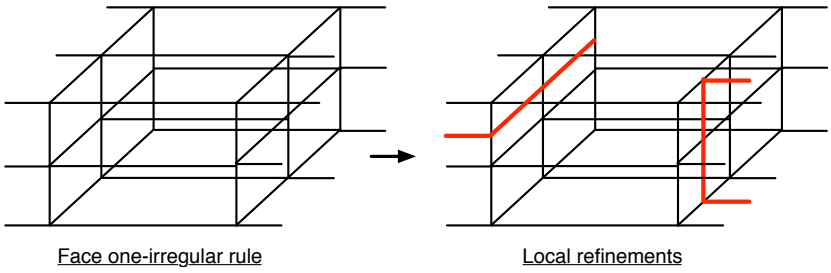
Figure 3.15: One-level reconstruction procedure of the element-to-nodes connectivity using refinement lookup arrays.

```
!> The routine recursively reconstructs nodal connectivities by
!! traversing the nodal trees
recursive subroutine elem_nodes(in::intr, out::nodes, out::orients)
  !! intr - a given interior node
  !! nodes, orients - reconstructed element-to-node connectivity

  if (is_root(intr)) then
    !! the recursion is terminated at a root (initial mesh element)
    call copy_nodes_from_initial_mesh(intr, nodes, orients)
  else
    call elem_nodes(intr.parent, nodes_temp, orients_temp)
    call reconstruct_elem_nodes(nodes_temp, orients_temp,
                                intr, nodes, orients)
  end if
end subroutine elem_nodes
```

Figure 3.16: Recursive reconstruction procedure of the element-to-nodes connectivity.



(a) Deadlock encountered during local mesh refinements



(b) Face one-irregular rule is enforced

Figure 3.17: The face one-irregular rule enforces to refine the element in the middle first to avoid a deadlock during the mesh adaptation process.

anisotropic refinements can subdivide a face node in such a way that no 3D refinement is available for neighboring elements. For example, consider a case depicted in Fig. 3.17a. The given mesh is admissible and one-irregular; two small face nodes on each side are constrained by the big face node associated with the element in the middle. Next, elements on both sides are locally refined as depicted in the figure. Consequently, the refined face nodes are now doubly constrained by the big face node. To recover the one-irregularity of the mesh, the element in the middle should be appropriately refined. However, no refinement can cope with such exotic refinement patterns of the face node. As a result, the mesh adaptation process reaches a deadlock.

A natural way to overcome this type of deadlocks is to first refine the element in the middle. As a result, the face nodes in the middle become regular so that refinement will take place in selected elements. The idea behind this approach is to enforce the one-irregularity rule for the face nodes. This treatment is necessary to support anisotropic refinements. Unlike in the case of isotropic refinements, an arbitrary combination of anisotropic refinements on a face node may produce multiple constraints on a face node, which may result in the deadlock as depicted in Fig. 3.7.

Motivated by this discussion, we introduce the following local mesh refinement rules to support isotropic and anisotropic refinements together:

Upgrading rule. A refinement type assigned to an element is always upgraded to accommodate existing refinements of face nodes.

Face one-irregular rule. No element is refined unless all its face nodes are active.

The first rule upgrades a given refinement flag by accommodating existing refinements of face nodes. This is easily implemented as the element refinement is decomposed into a set of orchestrated node refinements, where each node owns a separate hierarchy.

The face one-irregular rule creates a list of elements linked via constraining (inactive) face nodes. A series of mesh refinements is recursively performed on the list of elements to resolve the constraints on the face nodes. After all face nodes connected to a target element become regular, the target element is refined.

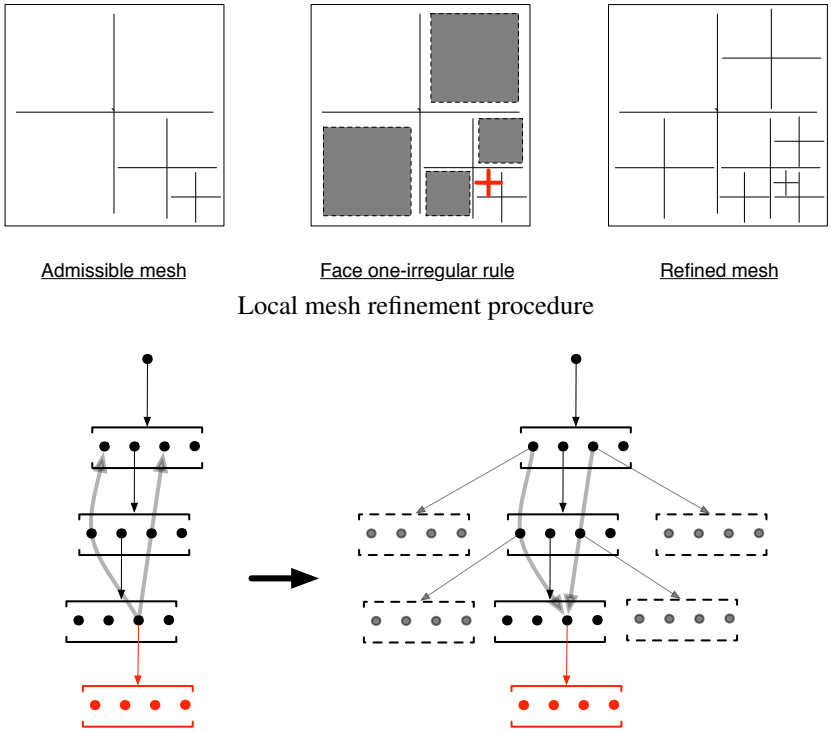


Figure 3.18: A list of elements connected through constrained face nodes refined observing the face one-irregular rule.

By enforcing these mesh refinement rules, the local refinement procedure is *deadlock free*. For simplicity, we explain this with 2D examples. Here, a face node is analogous to an edge node. As illustrated in Fig. 3.18, the face one-irregular rule refines a list of elements connected via constraining edges. Given the admissible mesh, a target element (red color) is selected and this refinement introduces doubly constrained nodes. Next, the face one-irregular rule gathers a list of elements; they are recursively refined in a Last-In First-Out (LIFO) manner.

As our second example, we consider a case that the face one-irregular rule

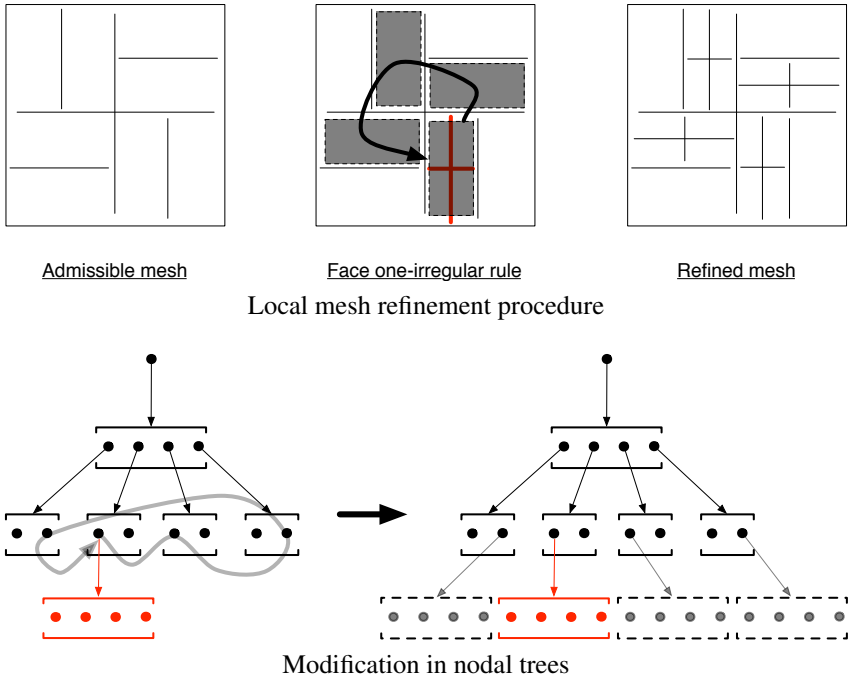


Figure 3.19: Anisotropic refinements may lead to a circular list of elements to keep the face one-irregular rule.

creates a circular list of elements. As depicted in Fig. 3.19, the face one-irregular rule creates a circular list of elements; the algorithm visits an element that is already kept in the list. When this happens, the algorithm start to refine elements in the list to avoid an unintentional infinite loop. The resulting mesh is still admissible.

From the perspective of the tree representation, it is easier to see that the local algorithms are deadlock free. Consider tree traversals illustrated in both Fig. 3.18 and Fig. 3.19. If the face one-irregular rule is seen in nodal trees, the algorithm collects a list of elements via traversing upward in the tree or horizontally (possibly circular) within the same hierarchy level. The first case is natural as nodes

```
!> The routine performs local mesh refinements keeping one-irregular
!! mesh rule on face nodes
subroutine refine(in::intr, in::refinement)
  !! intr - a given interior node to be refined
  !! refinement - refinement pattern to be applied

  type(stack_container) :: stack

  !! Step 1: push elements connected through constrained nodes
  stack.push_back(intr, refinement)
  intr.visited = .true.
  do while (.true.)
    !! pick the last element from the stack
    current = stack.back()
    call elem_nodes(current.intr, nodes, orients)

    !! check if all face nodes are unconstrained
    all_faces_unconstrained = .true.
    forall (iface=1:NFACE(current.intr))
      if (is_constrained(face_node(nodes,iface)) then
        neighbor = find_neighbor(current.intr, iface)
        if (neighbor.visited == .true.) then
          !! do not visit the element in the stack
          exit
        else
          !! if constrained, the neighboring element is added with
          !! a relevant refinement kind which is decided to resolve the
          !! face irregularity
          stack.push_back(neighbor, HP3D_DECIDE)
          neighbor.visited = .true.
          all_faces_unconstrained = .false.
        end if
      end if
    end forall

    !! exit condition to terminate the loop
    if (all_faces_unconstrained) then
      exit
    end
  end do

  !! Step 2: pop back an element from the stack and refine it
  do while (is_empty(stack))
    current = stack.pop_back()
    call break(current.intr, current.refinement)
    current.intr.visited = .false.
  end do
end subroutine refine
```

Figure 3.20: Pseudo code that describes the local refinement procedure.

are hierarchically constrained. The latter case is due to the presence of anisotropic refinements. An observation is that the algorithm never traverse downward in the tree; for both cases, the tree traversal is terminated within a finite number of iterations. This implies that the scheme is deadlock free. A detailed procedure for the local refinement algorithms is given in Fig. 3.20. This procedure has been verified through extensive numerical tests.

3.4.3 Global mesh closure

During the local refinements, selected elements are refined respecting the face one-irregular rule and the upgrading rule. Then, the resulting mesh may contain arbitrary-level hanging nodes on edges or vertices.

As an edge is always broken by half, a deadlock problem like one caused by unmatched face refinements is not encountered. One solution is to perform additional (unwanted) refinements to recover the mesh one-irregularity, called *mesh closure* [5]. The closure repeats the following three global-level operations until it recovers the mesh one-irregularity condition:

1. *Activate.* In this phase, we sweep all edges and vertices to count the number of applied constraints.

2. *Mark.* Next, the elements including double constraints on edges and vertices are marked.

[5]Note that if arbitrary-level hanging nodes for edges and vertices are allowed, this procedure is unnecessary. For 2D meshes and 3D structured meshes, there exist techniques supporting arbitrary-level hanging nodes [81, 89]; however, no implementation is found for 3D hybrid meshes.

3. *Refine.* Marked elements are refined to recover the mesh one-irregularity.

The closure is terminated in a finite number of iterations (in the worst case, we may converge to a uniformly refined mesh). At present though, we cannot provide a formal proof that the described algorithms are deadlock free. The claim has been verified through extensive numerical test involving randomly defined refinements resulting in hybrid meshes with millions of elements.

3.5 Construction of global basis functions

Global basis functions are constructed by gluing (or assembling) element shape functions with appropriate continuity conditions. To provide correct matching conditions at the inter-element boundaries, the element shape functions should be modified to account for the global orientations. For 2D elements, such modifications are easily made by applying a sign factor to the odd functions. For a detailed description, we refer to a book written by Demkowicz [26], pages 169–172.

For 3D elements, the orientation problem is more complicated. In particular, gluing triangular faces is a non-trivial task as the local coordinate system is not rotationally symmetric. This implies that a local shape function associated with a face node may need to be related to all face shape functions that belong to the neighboring element. In other words, one can say that the face shape function is rather interpolated by the corresponding all face shape functions in the neighboring element.

Instead, we can resolve this orientation problem by constructing the ele-

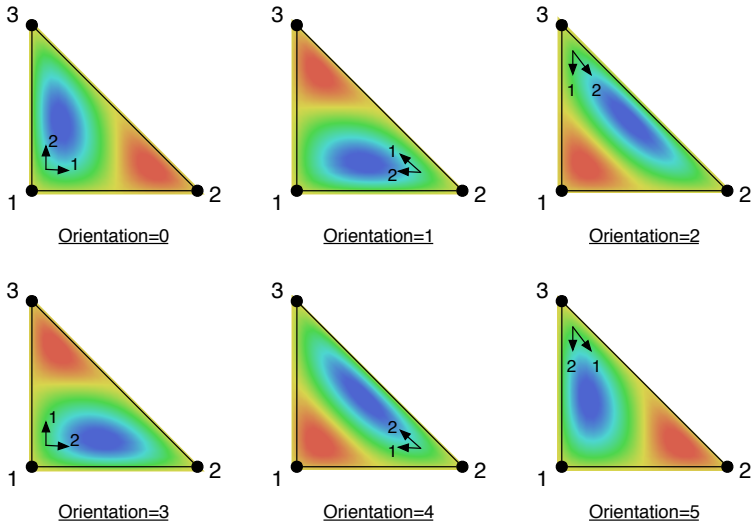


Figure 3.21: Examples of the orientation embedded shape functions. The first face bubble shape function corresponding to $p = 4$ is illustrated for different orientations.

ment shape functions in such a way that they automatically conform to the global edge and face orientations, see an example depicted in Fig. 3.21. The procedure of constructing the *orientations embedded shape functions* consists of three steps:

Step 1: Construction of edge or face shape function in the (global) edge or face coordinates.

Step 2: Change of coordinates from global-to-local as implied by the element coordinates and edge and face parameterizations defined in the abstractly defined element. This is where the information about the nodes orientations is used.

Step 3: Extension of the shape function to the whole element using appropriate blending functions.

For a more detailed description of the construction, we refer to [36, 80, 83].

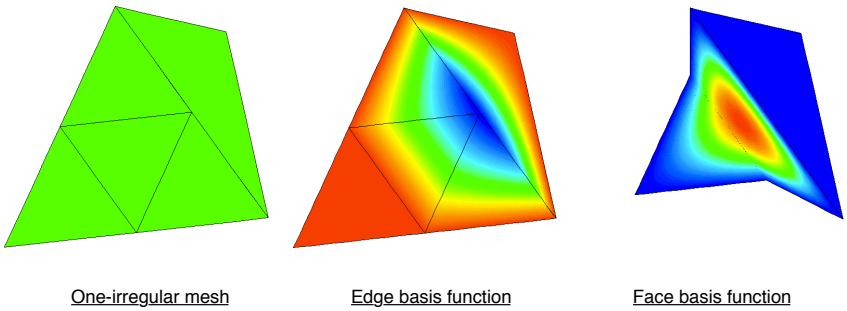


Figure 3.22: Constrained approximation on irregular tetrahedral meshes.

As the global orientations are controlled in the construction of the local shape functions, the assembly procedure simplifies to the one for classical FEs. This approach is more advantageous when implementing the constrained approximation.

The constrained approximation [25, 75] is introduced to obtain conformal global basis functions on irregular meshes, which are characterized by the presence of hanging nodes. To preserve proper continuity conditions across the inter-element boundaries, DOFs on hanging nodes must be expressed in terms of linear combinations of active (constraining) DOFs.

A major challenge again comes from the inconsistency of local and global coordinates for faces. For unstructured tetrahedral elements, the element local coordinates of refined elements can be arbitrarily set (see Fig. 3.9), which significantly increases the complexity of enforcing the continuity across the constrained faces. Probably, due to such technical difficulties, to our best knowledge, we could not find any implementation.

In our implementation (see Fig. 3.22), the complexity is considerably reduced as we construct local shape functions consistently with the global orienta-

tions. Remember that orientations are explicitly set up only for initial mesh elements, and the coordinates system for refined nodes are always inherited from their ancestors (initial mesh). Hence, we only need to build the constraint matrices corresponding to the cases depicted in Fig. 3.7 [6]. More detailed description is much too technical and go beyond the scope of this exposition.

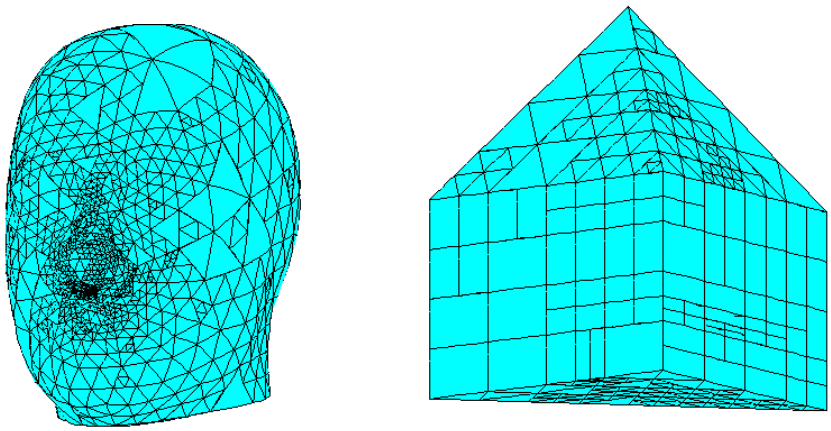
3.6 Code verification

The developed code has been cross-verified in collaboration with P. Gatto, who researched bone conduction of sound in the human head [35].

Deadlock free local mesh refinements. As claimed in Section 3.4, our local mesh refinement procedure should not incur a deadlock. To verify our conjecture, we performed a "torture test" that repeats local mesh refinements on randomly selected elements with arbitrary refinement flags. Figure 3.23 illustrates examples of randomly refined meshes after 20 iterations of the test. The test increased the number of elements up to a million starting with a minimum number of initial mesh elements to represent the geometry. Each refinement step was accompanied with a number of consistency checks. For instance, element-to-face and face-to-element [7]. connectivities (including orientations) were compared against each other.

[6] Quadrilateral face modes are tensorial and constraint matrices are dynamically formed by using the 1D constraint matrix.

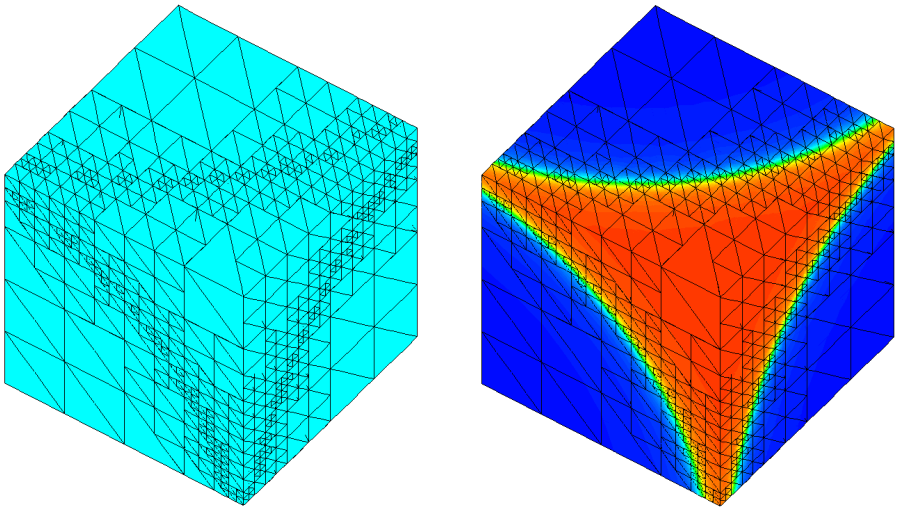
[7] One of several algorithms based on nodal trees that we have not discussed.



Tetrahedral mesh Hybrid mesh (prisms + tets)

Figure 3.23: Random mesh refinement test.

Verification of mesh generation. As described earlier, nodal geometry DOFs are determined by interpolating exact element geometry maps. The interpolation is performed in the reference GMP domain and has been verified by computing p- and h-convergence rates. Some of the geometrical parameterizations are of limited regularity and experience singularities in higher derivatives. As the boundary-value-problems are effectively solved in the reference domain with polynomial discretizations (see [27], pages 101–104 for an extensive discussion of the issue), computing interpolation h-convergence rates for the geometry maps not only helps to verify that the code is bug free but provides also a valuable information about the regularity of the geometry maps. As we solve for the composition of the element map and solution defined in the physical domain, irregular parameterizations may induce low regularity of the solution in the reference domain and overall low convergence rates. We discuss this later in Chapter 6.



(a) Locally refined mesh (b) Solution

Figure 3.24: Locally refined mesh to capture a shock in the middle.

Verification with manufactured solutions. Two classes of verifications have been performed. The first one deals with manufactured polynomial solutions defined on meshes with affine elements. With sufficiently high polynomial degree, these solutions have been reproduced with a machine precision including one-irregular meshes, verifying also constrained approximation routines. A second class of verification tests included more complicated manufactured solutions and computation of h-convergence rates for adaptive meshes. It is well known that h-refinements should restore asymptotically optimal h-convergence rates and checking this is part of the verification test.

An example of such a convergence test is illustrated in Fig. 3.24. We performed an H^1-projection test using the manufactured solution: $u(r) = \tan^{-1}(\alpha(r - r_0))$ with a parameter $\alpha = 60$ and $r_0 = \sqrt{3}$, where $r = |\mathbf{x} + (0.25, 0.25, 0.25)|$. The

solution involves a sharp layer inside of a unit cube domain. We used the standard greedy adaptive refinement strategy using element contributions to the global H^1 error (squared). Figure 3.24 shows the locally refined mesh to capture the shock.

3.7 Summary

In this chapter, we presented `hp3d`, a general *hp*-FE code, that is designed to support exact sequence elements for multi-physics problems. The code supports hybrid meshes including the elements of all shapes (*i.e.*, hexahedron, tetrahedron, prism, and pyramid), which allows us to describe a complex geometry.

From the implementation perspective, the code stores only two object arrays; `ELEMS` stores the full spatial connectivity of a given initial mesh; `NODES` stores a family of nodal trees that records *h*-refinements. With generalization of an element as a set of nodes, major subroutines in the code is written in an "object-oriented" fashion.

A particular interest is in the development of a general mesh refinement scheme including various isotropic and anisotropic refinements in hybrid meshes that includes the elements of all shapes. The proposed refinement scheme completes in two steps allowing one-irregular hanging nodes. In the first step, local mesh refinements are applied to marked elements. During the local refinement procedure, the algorithm refines a list of elements that are connected through constrained face nodes, called the face one-irregular rule. This local algorithm is necessary to avoid a "deadlock" which possibly occur with anisotropic refinements. At this stage, the mesh may include arbitrary-level hanging nodes on edges and vertices. In the

second step, the global mesh closure performs additional refinements to recover the mesh one-irregularity.

Since we allow one-irregular meshes, DOFs on hanging nodes should be appropriately constrained by regular (active) nodes. For ease of implementation, we introduce a new orientation embedding technique in shape functions; shape functions are always constructed with respect to the global orientations. This idea significantly simplifies the assembly procedure and constrained approximation.

Chapter 4

Unassembled HyperMatrix Solver

In this chapter, we discuss an efficient sparse direct solver targeting linear systems of equations derived specifically from the *hp*-Finite Element (FE) method. The following two aspects are extensively used in designing of this new sparse direct solver:

- Sparse linear systems are locally updated during mesh adaptation. As the mesh is locally refined, the solver can preserve the partial factors previously computed and reuse them for solving the current system of equations [14].
- Element matrices derived from an *hp*-discretization include fairly large dense subblocks characterized by the order of approximation p. This implies that all operations are essentially dense, which allows highly efficient level 3 BLAS functions [29] to be utilized.

Compared to a conventional black-box solver, the proposed solver gains additional performance from the *hp*-workflow interface by exploiting application information inherited from the *hp*-FE method.

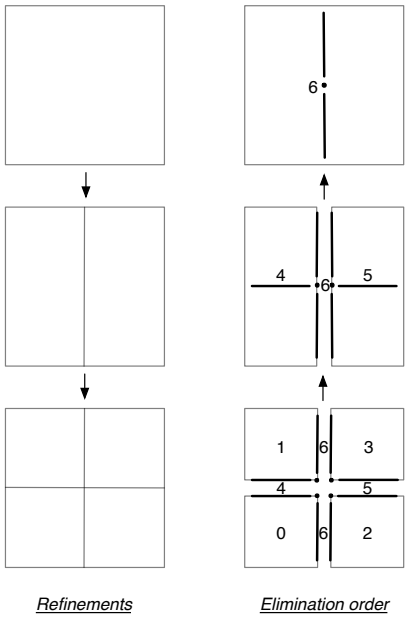


Figure 4.1: Applied mesh refinements and its induced elimination ordering.

4.1 Source of sparse matrices: the *hp*-Finite Element Method

Adaptive workflow. During the mesh adaptation process, the same problem is repeatedly generated and solved with local mesh refinements; the next system of equations is locally updated from the previous one. A drawback in using the standard black-box approach is that element matrices are recomputed even for the elements inherited from the previous mesh. Similar redundant computations are also performed in the linear solver. Consequently, a black-box solver may not achieve the best overall application performance when it only focuses on an individual subproblem.

We presented a new approach, called Unassembled HyperMatrix (UHM) solver [14], to solve sparse matrix problems with local *hp*-refinements. The solver

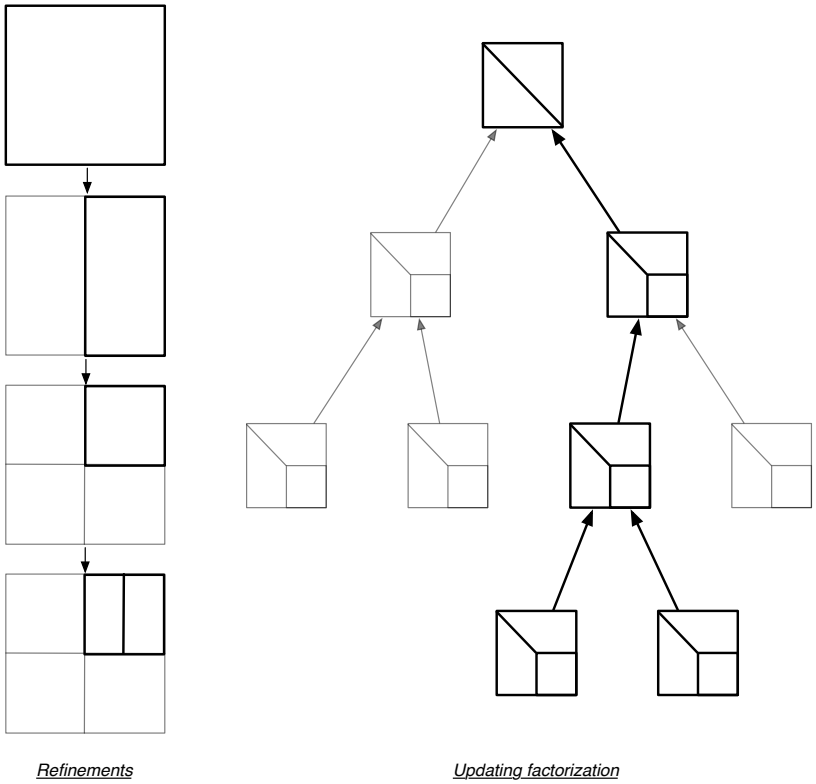
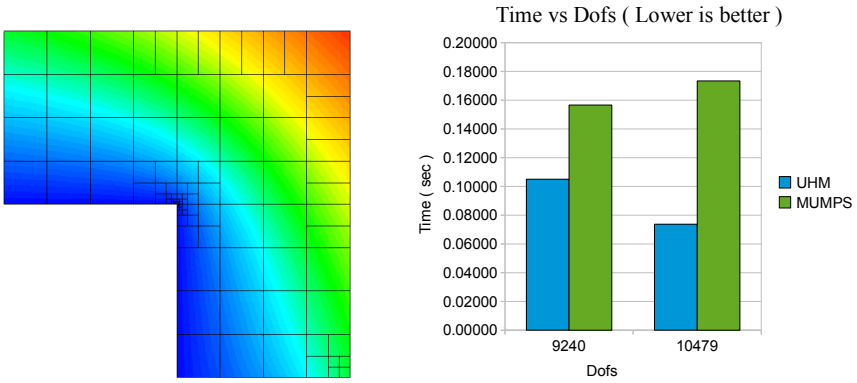


Figure 4.2: Updating factorization with local mesh refinements. In the last adaptive iteration, element matrices and their partial factors previously computed (grey lines) are reused and element matrices (black line) affected by the applied local refinement are updated.

is designed based on the multifrontal method. However, the solver is distinct from other sparse solvers in that it has an ability to construct an elimination order from the applied mesh refinements as illustrated in Fig. 4.1. This induced elimination order provides an opportunity to reuse the partial factors that was previously computed. As depicted in Fig. 4.2, the solver does not factorize the entire system of equations from scratch but reuses some of partial factors. For instance, the problem depicted in Fig. 4.3 has a singularity at the re-entrant corner, and the *hp*-adaptivity refines the



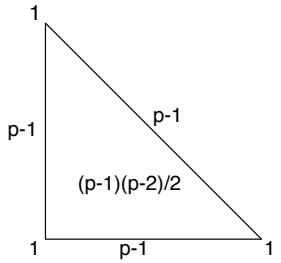
(a) L-shape solution (b) Updating factorization

Figure 4.3: Time complexity of the updating factorization in the mesh adaptation procedure.

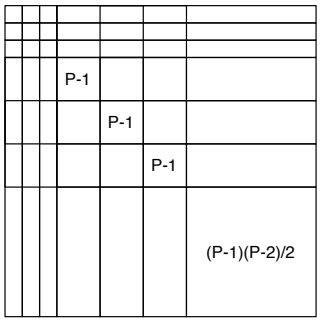
mesh around the corner. One can see that a fairly large amount of element matrix computations and their partial factors can be reused in the next adaptive iteration. The graph in the figure shows that the updating factorization gives us considerable cost saving cost compared to MUMPS that factorize the matrix from scratch. We described details of this approach in [14].

Supernodes. A characteristic of the hp-discretization is that multiple DOFs can be grouped in a topological node, as illustrated in Fig. 4.4a. This in turn gives a shape of an element matrix depicted in Fig. 4.4b. By assembling element matrices, a global system of equations is formed. As seen in Fig. 4.5, higher p gives a block sparse structure, which is denser than one derived from a linear order of approximation.

A sparse system derived from an hp-discretization can be described in terms

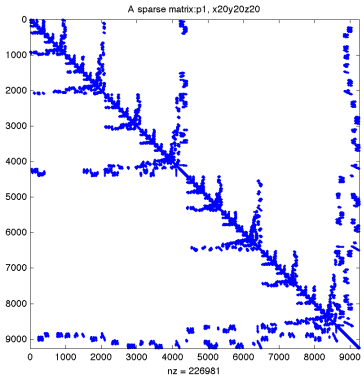


(a) High order element

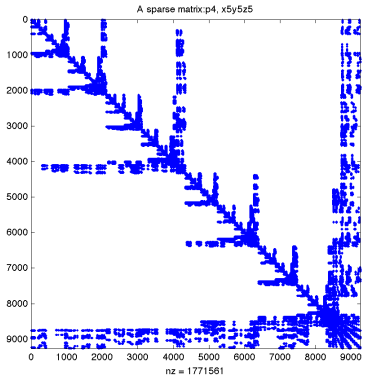


(b) Element matrix

Figure 4.4: An element matrix structure derived from the high order discretization with a polynomial order of approximation p.



(a) $p = 1$, nz = 226,981



(b) $p = 4$, nz = 1,771,561

Figure 4.5: Characteristic sparsity patterns for $p = 1$ and $p = 4$ keeping the same system DOFs.

of topological nodes and associated DOFs rather than individual variables. Formally, we can create a quotient graph of unknowns by dividing out the equivalence class of basis functions associated with the same topological node. This leads an efficient solver since the graph is considerably smaller while as a bonus presenting

an increased opportunity for using Level 3 BLAS operations [29]. In contrast, the black-box interface of a general purpose sparse direct solver does not recognize well the presence of such supernodes *a priori*.

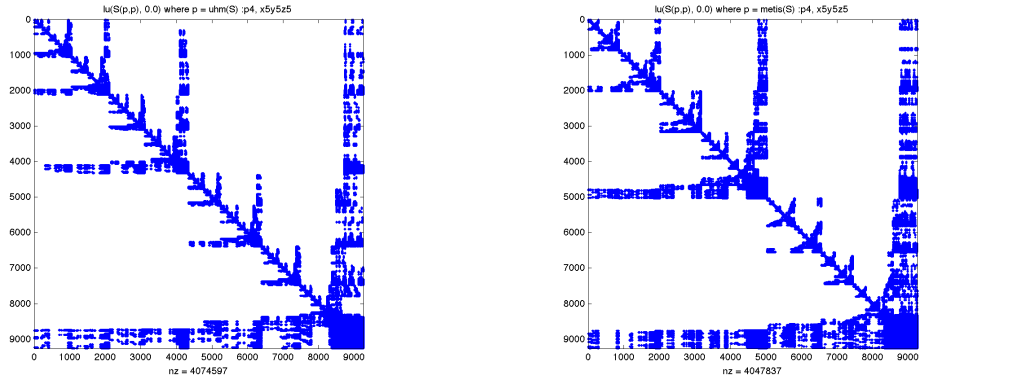
4.2 Factorization algorithms

A general procedure for direct methods consists of four phases: ordering, analysis, factorization, and forward/backward substitution. Our factorization scheme has the same structure, but several aspects are different from the general sparse direct solver as we have a specific target application of the *hp*-FE method.

4.2.1 Ordering strategy

In our ordering phase we construct a partial order of the elements and subdomains. For an initial *hp*-mesh given without a hierarchy, a refinement tree is virtually constructed *a posteriori*. This gives us a balanced tree where all elements are recursively defined as refinements from a (fictitious) top element. From this tree structure we obtain a partial ordering of elements; the factorization tasks will be forced to obey this ordering. We also emphasize that a full linear ordering of the unknowns is never constructed. All unknowns inside of an element matrix in a level are treated together as a block; in between levels we observe a partial ordering that the coarse subdomains can only be eliminated after the finer subdomains.

The ordering strategies are very different; our solver mirrors the *hp*-mesh structure and preserves inherent dense subblocks characterized by the order of approximation p. In contrast, the standard black-box interface loses the block struc-



(a) UHM (mesh mirroring), nz = 4,074,597 (b) Node-based nested dissection, nz = 4,047,837

Figure 4.6: The left figure shows fills computed by the UHM solver, which keeps the given block structures ($p = 4$) of the matrix. On the other hand, the figure on the right shows fills computed by node-based nested dissection ordering.

ture of the hp-mesh and such information is imperfectly reconstructed. In Fig. 4.6, we compare nonzero fill patterns after symbolic factorization respectively obtained from our solver and Metis [46] node-based nested dissection ordering. One can clearly see the difference between two sparsity patterns; however, the two approaches report a similar number of nonzeros and floating point operations. However, the different ordering may directly impact the performance of the numerical factorization. For this particular case, the scattered fill distribution constructed by Metis may lead to less efficient use of level 3 BLAS functions.

4.2.2 Unassembled matrix factorization

We store matrices in an unassembled form: (dense) matrices corresponding to elements are stored on leaves in the tree, where inter-element boundary variables

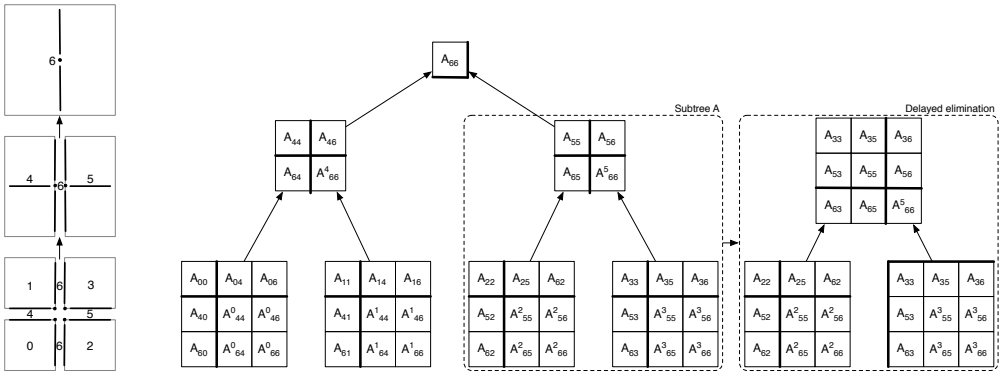


Figure 4.7: An assembly tree can be organized by recursively coarsening the *hp*-mesh. The constructed tree can also be dynamically modified to control a large element growth.

are remained unassembled. This strategy is illustrated in Fig. 4.7. The procedure is recursively driven by a partial order tree traversal. More formally we describe this recursion as follows.

1. *Subassembly.* An element matrix on any but the tree level is assembled from its children:

$$A := \text{assemble}\left(A_{BR}^{left}, A_{BR}^{right}\right),$$

where A_{BR}^{left} and A_{BR}^{right} represent the Schur complements from the children.

2. *Identification of interior variables.* The element matrix is permuted and conformally partitioned into quadrants with square diagonal blocks separated by their topological relation: the fully assembled internal variables and the boundary variables:

$$A \to \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right).$$

3. *Partial LU factorization with pivoting of a block A_{TL}.* A_{TL} contains the fully assembled nodes of an element and can therefore be eliminated. Block LU factorization is applied to the matrix A where pivots are selected within the submatrix A_{TL}:

$$\left(\begin{array}{c|c} P & 0 \\ \hline 0 & I \end{array}\right)\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) = \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right)\left(\begin{array}{c|c} U_{TL} & U_{TR} \\ \hline 0 & U_{BR} \end{array}\right)$$
$$\rightarrow \left(\begin{array}{c|c} \hat{A}_{TL} & \hat{A}_{TR} \\ \hline \hat{A}_{BL} & \hat{A}_{BR} \end{array}\right),$$

where L_{TL} and L_{BR} are normalized to have unit diagonal entries, and U_{TL} and U_{BR} are upper triangular matrices. Factors are computed in-place and overwritten on the same submatrices. By equating corresponding submatrices on the left and right, the partial elimination proceed as follows:

   - Interior variables within the submatrix A_{TL} are eliminated by a standard right-looking LU algorithm, giving the factors $\hat{A}_{TL} = \{L_{TL} \backslash U_{TL}\}$.
   - Next, submatrices A_{TR} and A_{BL} are respectively updated into $\hat{A}_{TR} := L_{TL}^{-1} A_{TR}$ and $\hat{A}_{BL} := A_{BL} U_{TL}^{-1}$.
   - Submatrix A_{BR} is updated into $\hat{A}_{BR} := A_{BR} - \hat{A}_{BL}\hat{A}_{TR}$, which yields a Schur complement to update its parent.

4. *Stability check.* After this partial elimination, we check the element growth of $\hat{A}$ to ensure the factorization is stable with a given threshold α:

$$\max\left|\hat{A}_{ij}\right| < \alpha \cdot \max\left|A_{ij}\right|,$$

where α is a threshold value to monitor excessive element growth. In general, the α is a function of a prescribed constant value and size of the element

matrix. For instance, we use the following criterion:

$$\alpha = (1.0 + u_0^{-1}) \cdot \text{size}(A_{TL}),$$

where the criterion is slightly modified from [30]. In the case that unacceptable element growth is encountered, matrices are re-assembled and corresponding eliminations for the interior variables are delayed to their parent, see Fig. 4.7.

The recursive factorization finishes at the root, where A_{BR} is an empty matrix.

4.2.3 Stability check

In the course of multifrontal factorization, large element growth leads to numerical instability [33] and should be managed in a proper manner keeping fills as small as possible [55]. In principle, we use standard partial pivoting (row exchange) within the elimination block matrix. After each element matrix is factorized, we check the element growth; if unacceptable element growth is detected, child Schur complements are re-assembled and elimination of the nodes are delayed to its parent. Thus, the solver can examine and select a potential pivot from a relatively large column vector. We use this approach to preserve the block structures derived from the hp-discretization at the factorization phase.

Figure 4.8 compares the element growth during unassembled factorization with different stability thresholds. In practice, as the size of UHMs increases at the upper-level hierarchy, partial pivoting within its elimination block become stable for such cases. On the other hand, when delayed pivots occur for small sized element

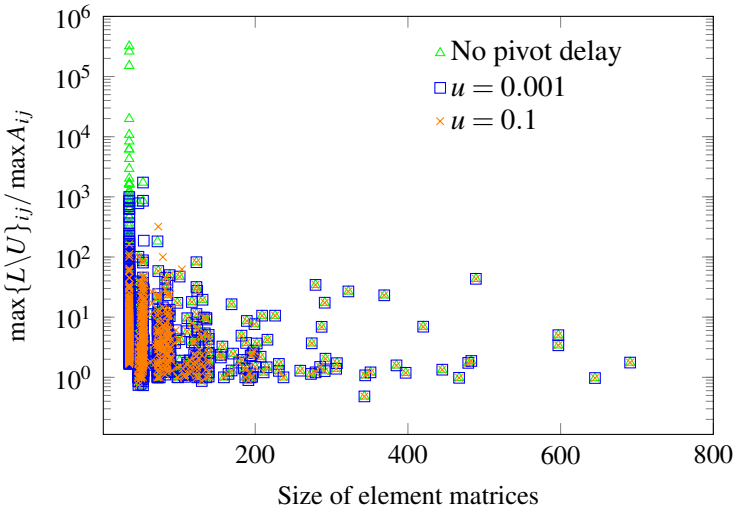


Figure 4.8: Element growth of dense subproblems for different pivot thresholds.

matrices, the overhead in handling (recomputing) those pivots slightly increases the factorization cost.

4.2.4 Algorithms-by-blocks

The multifrontal method has a natural parallelism from its recursive tree traversal; tasks on separate branches can be processed simultaneously. However, the task parallelism decreases as the factorization is closer to the root. Further parallelism for the dense blocks nearby the root is necessary to improve the efficiency of the parallel factorization.

Further task-level parallelism can be pursued by organizing a matrix by blocks (submatrices) [40, 57, 93]. For processing these blocks, we use so-called *algorithms-by-blocks* [16, 19, 74], which reformulate dense matrix computations algorithms in terms of blocks. For example, consider a matrix is partitioned into an

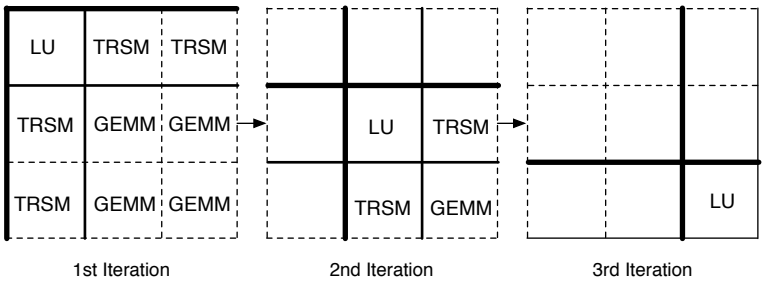


Figure 4.9: Algorithm-by-blocks of LU factorization without pivoting on a 3×3 block matrix.

$N \times N$ matrix of blocks:

$$A = \begin{pmatrix} A^{(0,0)} & A^{(0,1)} & \dots & A^{(0,N-1)} \\ A^{(1,0)} & A^{(1,1)} & \dots & A^{(1,N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ A^{(N-1,0)} & A^{(N-1,1)} & \dots & A^{(N-1,N-1)} \end{pmatrix}$$

In the first iteration of LU factorization without pivoting,

- $A^{(0,0)}$ is factored,
- for $i = 1, \dots, N-1$ block $A^{(i,0)}$ is overwritten by $A^{(i,0)}U^{(0,0)-1}$; this triangular solve with multiple right-hand sides is done with the BLAS routine TRSM;
- for $k = 1, \dots N-1$ block $A^{(0,k)}$ is overwritten by $L^{(0,0)-1}A^{(0,k)}$; this is a triangular solve with multiple right-hand sides, executed by TRSM, and
- for $i,k = 1, \dots, N-1$ blocks $A^{(i,k)}$ are overwritten by $A^{(i,k)} - A^{(i,0)}A^{(0,k)}$; requires a matrix-matrix multiplication using the BLAS routine GEMM.

This and subsequent iterations are illustrated in Fig. 4.9.

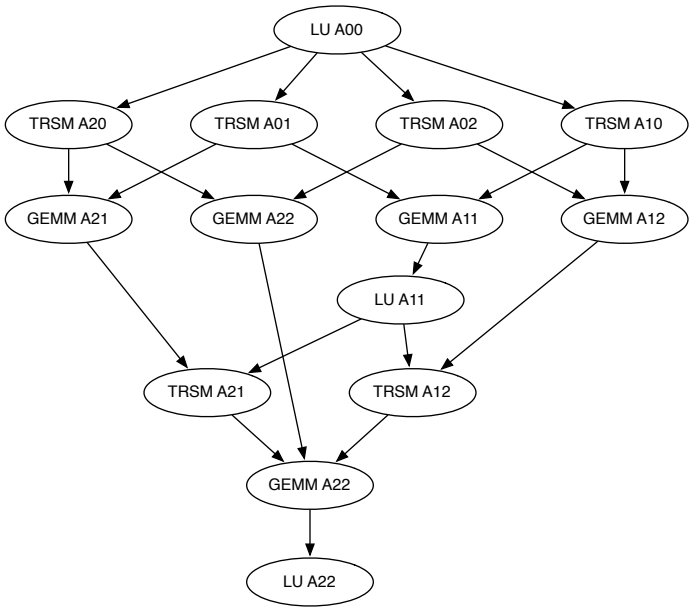


Figure 4.10: An example of a DAG of tasks produced by applying the algorithm-by-blocks of LU factorization without pivoting.

Fine-grained tasks are generated via algorithms-by-blocks. After task dependencies are analyzed, tasks are scheduled asynchronously. This leads to highly efficient task parallelism on modern multi-core architectures. An example of this approach for the LU factorization without pivoting is shown in Fig. 4.10.

4.3 Summary

In this chapter, we presented the unassembled factorization scheme and its efficient workflow interface to *hp*-adaptive FE methods. In particular, our solver utilizes the following information inherited from *hp*-refinements:

- an elimination ordering can be derived from *h*-refinements;

- higher order approximation gives supernodal information *a priori*.

To incorporate this application information, we store unassembled element matrices in a tree structure and perform the factorization. Nodes are only assembled when they are eliminated and it produces a generalized element matrix in the next level of the tree. Our solver has an ability to dynamically insert/delete element matrices resulting from hp-refinements. With an elimination ordering induced from h-refinements, our solver reuses the partial factors that were previously computed [14].

Representing a global sparse system of equations, we construct a quotient graph that mirrors the hp-mesh in terms of topological nodes and their DOFs. Our graph representation is considerably smaller than one constructed with individual variables. This condensed graph is beneficial as it preserve the block structures in the sparse matrix, which allows efficient Level 3 BLAS operations.

Finally, we briefly give an overview of algorithms-by-blocks and their use in our solver.

Chapter 5

High Performance Computing

In this chapter, we present an approach to multi-level task scheduling that matches the two-level parallelism in the multifrontal factorization. Our solution is based on Directed Acyclic Graph (DAG)-based task scheduling. Unlike the classic fork-join parallel model, this approach enables efficient asynchronous task scheduling. Our task scheduler is unique in that multiple instances of the task scheduler are invoked for solving subproblems that are encountered during the multifrontal factorization.

5.1 Hierarchical task scheduling

In the course of a sparse factorization, as depicted in Fig. 5.1, we have two oppositely behaving types of parallelism as the factorization progresses from the leaves to the root of the tree:

- a decreasing amount of task parallelism;
- an increasing opportunity for parallelism inside the blocks as their sizes grow.

Now, the question is how to exploit the two-level parallelism in harmony to extract near-optimal utilization from multi-core architectures while avoiding exces-

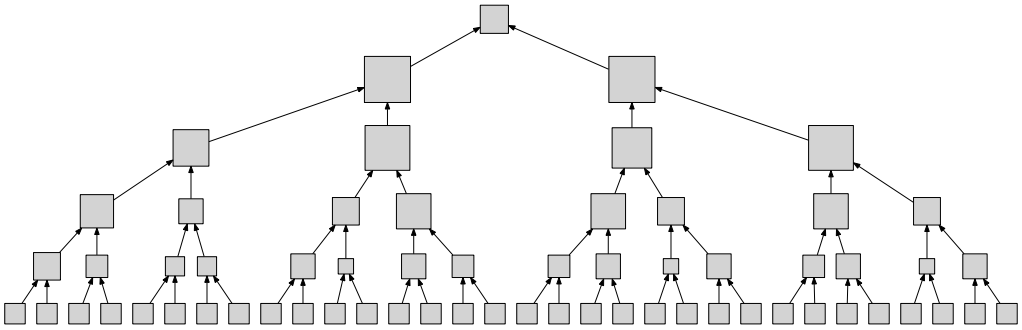


Figure 5.1: Irregular coarse grain tasks resulting from the multifrontal factorization characterized by its assembly tree.

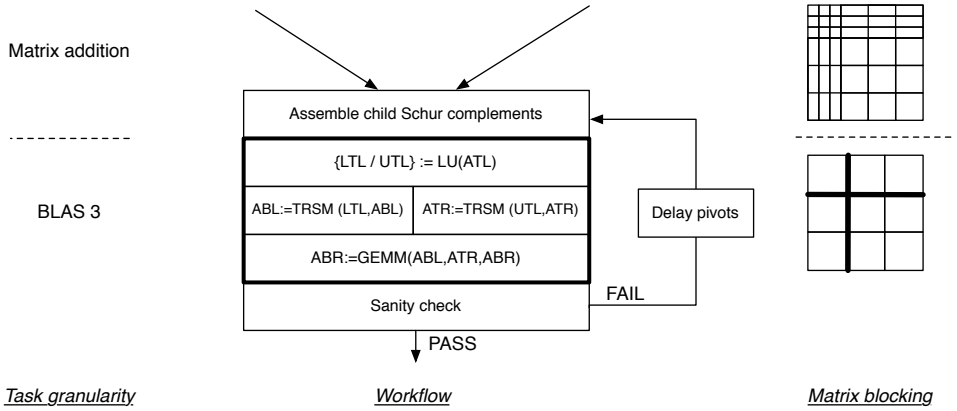


Figure 5.2: The workflow of the tree-level task associated with partial factorization of the element matrix.

sive complexity in implementation.

In our parallelization approach, coarse grain tasks are first generated by the post-order tree traversal along the assembly tree: each node in the tree becomes a task that handles an element matrix. These tree-level tasks are hierarchically related and a task cannot proceed until its children have been processed. We illustrate a tree-level task in Fig. 5.2. The workflow in a typical tree-level task consists of the

assembly of the Schur complement of the children factorization, which carries an $O(n^2)$ cost, followed by level 3 BLAS operations for the factorization, all of which are $O(n^3)$ in the element matrix size.

By applying algorithms-by-blocks in each tree-level task, a group of fine-grained tasks is generated and their dependencies are captured by a DAG; the entire factorization can then be represented by hierarchically related subgraphs as depicted in Fig. 5.3. Rather than unrolling all fine-grained tasks in a global DAG, we first schedule tasks encountered during tree-traversal. Those tree-level tasks are then scheduled with local DAG schedulers. In this way we *schedule schedulers*. Next, each group of fine-grained tasks is dispatched with local scheduling policies, which are guided by the associated local DAG. No global DAG is formed; thus, scheduling overhead is considerably reduced.

5.1.1 Parallel tree traversal

We use OpenMP to schedule all tasks generated in the factorization phase. We do not use an explicit data structure to reflect the schedule according to which tasks are to be executed. Instead, we rely on the OpenMP framework: by declaring tasks in the right execution order with OpenMP pragmas, they are entered into the OpenMP internal scheduler.

The mechanism we use is a part of the explicit task management that was added in OpenMP 3.0, released in 2008 [68]. The new task scheme includes two compiler directives:

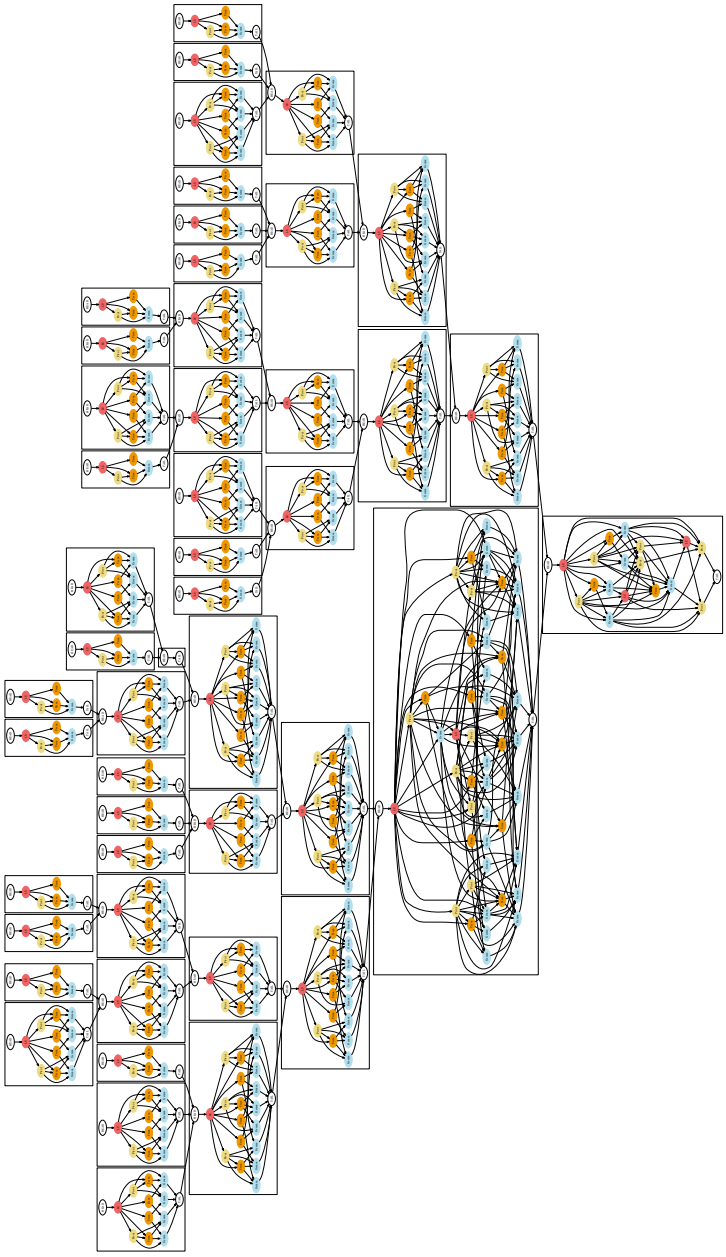


Figure 5.3: Illustration of two-level task scheduling.

```
// ** Sparse factorization via UHMs
int factorize_sparse_matrix(Tree::Node *root) {
  // begin with the root node
  post_order_task(root));
  return SUCCESS;
}

// ** Post-order tree traversal
int post_order_task(Tree::Node *me) {
  for (int i=0;i<me->get_n_children();++i) {
    // tree-level task generation for child tree-nodes
    #pragma omp task firstprivate(i)
    post_order_task(me->get_child(i));
  }

  // parent task is suspended
  #pragma omp taskwait

  // process the function (local scheduling)
  factorize_uhm(me);

  return SUCCESS;
}

// ** Partial factorization in UHM
int factorize_uhm(Tree::Node* nod) {
  // merge the Schur complements from child tree-nodes
  nod->assemble();

  // local DAG scheduling for LU factorization
  Scheduler s;

  // tasks are created using algorithms-by-blocks
  // and they are associated with a local scheduler
  create_lu_tasks(nod->ATL, s);
  create_trsm_tasks(nod->ATL, nod->ABL, s);
  create_trsm_tasks(nod->ATL, nod->ATR, s);
  create_gemm_tasks(nod->ABL, nod->ATR, nod->ABR, s);

  // parallel execution of tasks in a DFS-like manner
  s.flush();

  // monitoring element growth;
  // if necessary, UHM is re-assembled and eliminations are delayed
  nod->stability_check();

  return SUCCESS;
}
```

Figure 5.4: Recursive generation of tree-level tasks.

- `#pragma omp task` creates a new task
- `#pragma omp taskwait` is used to synchronize invoked (nested) tasks.

Importantly, a task can recursively create descendent tasks inside the task. When a task spawns descendent tasks, `#pragma omp taskwait` can suspend the task until those tasks are completed. For example, Fig. 5.4 outlines the parallel multifrontal factorization via parallel post-order tree traversal with OpenMP tasking. Invoked tasks are scheduled based on a Breadth First Search (BFS) order; tasks in the current recursion level are executed before their parent task is processed.

OpenMP utilizes thread pools to execute tasks; when a task is ready to execute, an idle thread picks it up to process the task. Hence, the programming burden for dispatching tasks is removed, and portable performance can be achieved in various architectures which support OpenMP standards.

5.1.2 Scheduling the block matrix operations

Using algorithms-by-blocks, we generate a list of fine-grained tasks, and these resulting tasks can be scheduled by an existing scheduler like SuperMatrix [19, 74] or PLASMA [16]. The problem here is that the number of tasks is very large, which causes excessive overhead in scheduling. Moreover, task sizes that encountered are very irregular; the grain size for assembling element matrices (a set of block additions) is much smaller than tasks generated from elimination process (level 3 BLAS). Scheduling such small sized tasks may incur more overhead rather than parallel performance. A further problem is that the task list is dynamic

0 LU	1 Tr	2 Tr	3 Tr	4 Tr	5 Gm	6 Gm	7 Gm	8 Gm	9 LU	10 Tr	11 Tr	12 Gm	13 LU
-	0 LU	0 LU	0 LU	0 LU	1 Tr	1 Tr	2 Tr	2 Tr	5 Gm	5 Gm	5 Gm	6 Gm	8 Gm
-	-	-	-	-	3 Tr	4 Tr	3 Tr	4 Tr	-	7 Gm	6 Gm	7 Gm	12 Gm
-	-	-	-	-	-	-	-	-	-	9 LU	9 LU	8 Gm	-
-	-	-	-	-	-	-	-	-	-	-	-	10 Tr	-
-	-	-	-	-	-	-	-	-	-	-	-	11 Tr	-

Table 5.1: Tasks are generated by applying algorithm-by-blocks of LU factorization without pivoting to a 3×3 block matrix. The first row represents a list of enqueued tasks; each task records a set of pointers to dependent tasks, which are listed in a corresponding column.

because of numerical pivoting, and these packages can not yet deal with that.

Adopting DAG-based task scheduling for matrix-level parallelism, we designed a new out-of-order task scheduler and implemented this using the OpenMP framework. As OpenMP is used for low-level binding of tasks to threads, we have no control over the specific order in which tasks are executed. Instead, our task scheduler invokes a list of tasks in parallel and enforces a partial order that observe dependencies among tasks. More specifically, our implementation is based on the following scheduling rules;

- no task is executed before all dependent tasks are executed;
- tasks are executed only once regardless of how many times they are invoked.

In addition, as these local schedulers are scheduled in the proper post-order in Fig. 5.4, we find that all tasks are executed in the right global order with only local analysis.

We explain our approach using the example of the dense LU factorization without pivoting. Consider LU factorization on a 3×3 block matrix in Fig. 4.9. A

```
// ** Invoke all tasks
int Scheduler::flush() {
  while (tasks_not_empty()) {
    open_window();                  // set a range of tasks for analysis
    analyze();                      // construct a DAG for those tasks
    execute();                      // execute tasks in parallel
  }
  tasks.clear();                    // clean-up task queue

  return SUCCESS;
}

// ** Execute tasks in an active window
int Scheduler::execute() {
  for (int i=_begin;i<_end;++i) {
    #pragma omp task firstprivate(i) // schedule fine-grained tasks using OpenMP
    this->tasklist.at(i).execute_once();
  }
  #pragma omp taskwait              // complete the execution of a set of tasks
  close_window();                   // close the window

  return SUCCESS;
}

// ** Tasking policies
int Task_::execute_once() {
  // return flag
  int r_val = SUCCESS;

  // atomic capture of the execution status
  int status = __sync_add_and_fetch(&this->_once_execute, 1);

  // Rule 2: execute the current task once
  if (status == 1) {
    // Rule 1: recursive calls on dependent tasks
    for (int i=0;i<this->get_n_dependency();++i) {
      #pragma omp task firstprivate(i)
      this->get_dependent_task(i)->execute_once();
    }
    // execute the current task after all dependent tasks are processed
    #pragma omp taskwait
    r_val = this->execute();
  }

  // yield the current thread until this task is completed
  while (this->_n_dependency != COMPLETED) {
    #pragma omp taskyield
  }
  return r_val;
}
```

Figure 5.5: A pseudo code that describes recursive task scheduling.

list of tasks is generated by the LU algorithm-by-blocks. After task dependencies are analyzed, tasks are organized in the scheduler as tabulated in Table 5.1. Details of our scheduler are given in Fig. 5.5. Notably, the `execute` invokes a list of tasks in the first row of the table and the first scheduling rule drives a recursion on the dependent tasks. For any given task in the list, OpenMP executes the task in an order that obeys the dependencies. For instance, `task 6` creates a recursive call stack:

```
Task 6 -> Task 1,4 -> Task 0.
```

This recursive process is naturally suited to the nested parallelism supported by the `omp task` pragma. However, the recursion can also invoke the same task multiple times. The second rule prevents this situation and enforces a task to be executed once by the first-reached thread. In this example, both `Task 1` and `Task 4` invoke `Task 0`, but `Task 0` is exclusively executed by the first-reached thread; the other encountered thread is redirected to other available tasks by `omp taskyield`. In this scheduling mechanism, the task table such as one depicted in Table 5.1 can be interpreted as scheduling hints which represent the concurrent tasks (in a row) and data locality (in a column):

- tasks in the first row can be independently executed in an arbitrary order, and
- dependent tasks in the same column can be tied to the current working thread in favor of reusing data.

Together with various work-queue models and scheduling policies studied in [31,

67, 82, 91], the current OpenMP implementation shows reasonable performance gains in both dense and sparse matrix factorization.

5.1.3 Memory usage in the tree traversal

During the post-order tree traversal, the solver dynamically creates/destroys temporary storage for Schur complements. In principle, the necessary memory usage for the factorization mostly depends on the applied fill-reducing ordering. However, actual memory usage in the parallel factorization also depends on the shape of the assembly tree and specific parallel implementation [38].

With regards to the memory usage in our implementation, the solver has the following characteristics:

- Our solver creates a virtual refinement tree (assembly tree) using a recursive bisection scheme on an *hp*-mesh; consequently, the constructed tree is well-balanced.
- During the recursive factorization, element matrices are dynamically created and destroyed by encountered threads. We also note that our solver neither pre-allocates nor over-allocates matrices before the factorization begins.

As depicted in Fig. 5.4, tasks are generated within a BFS manner to improve the level of concurrency. This approach naturally assigns threads to independent subtrees during the tree traversal. All necessary temporary storage is dynamically allocated by the encountered threads. This feature is well suited since, Non-Uniform Memory Access (NUMA) architectures by default, physical memory

Lonestar	
Processors	4x6 Intel Xeon E7540 2.0 GHz
Memory	1TB, 64x16GB, DDR3-1066MHz, 4x QPI
Cache	18 MB shared L3 cache, 256 KB L2 cache/core
Compiler & OpenMP	GNU 4.4.5
BLAS	Intel MKL 12.1
DLA	FLAME ver 10519
Peak performance	8.0 GFLOPS/core

Table 5.2: Specifications of the *Lonestar* large memory node at the Texas Advanced Computing Center (TACC).

is allocated on the local NUMA node in which the thread is running, following a "first-touch" rule.

Next, nested tasks created within a task are sent to a private task pool associated with a thread executing the current task and the current task is suspended by `omp taskwait`. OpenMP gives a priority to tasks in the thread-private pool. As a result, a high degree of data locality can be achieved in each thread.

5.2 Performance on a Symmetric Multi-Processing architecture

We compare the performance of the proposed solver against the state-of-the-art parallel sparse direct solvers MUltifrontal Massively Parallel sparse direct Solver (MUMPS) and PARDISO:

- MUMPS [5, 6] was developed for distributed architectures via Message Passing Interfaces (MPI) starting in 1996. For this comparison, MUMPS version 4.10.0 is interfaced to the Scalable Linear Algebra PACKage (ScaLAPACK) and Basic Linear Algebra Communication Subprograms (BLACS) provided

by Intel's Math Kernel Library (MKL) version 12.1. Single threaded BLAS and Linear Algebra PACKage (LAPACK) are interfaced with 24 MPI nodes.

- PARDISO [78,79] solver was developed for multi-core architectures in 2004, and is a part of MKL. For this comparison, we use version 12.1.

For all cases, we use a single node with 24 cores of *Lonestar*, a massively parallel architecture at the Texas Advanced Computing Center (TACC), which we describe in Table 5.2. Solution accuracy is not reported as all solvers produces similar relative residual.

Strong scalability.

We consider the speed-up by increasing the number of processing units for a fixed problem derived from a high order discretization ($p = 4$).

In this comparison, Metis version 4.0 [46] is commonly interfaced to reorder the sparse matrix. Detailed performance data is tabulated in Table 5.3 and Table 5.4; some observations are:

- While the UHM solver and MUMPS report the almost same number of FLoating Point Operations (FLOPs) in the factorization phase, PARDISO estimates almost twice that of the others.

- As for the utilization of multi-core resources, PARDISO and UHM solvers show higher performance than MUMPS.

	Total cost: 3476 GFLOP		
Cores	Time (sec)	GFLOP/sec	Memory (GB)
1	473.65	6.84	7.37
2	257.36	13.43	8.25
4	189.77	17.42	8.14
8	93.02	35.03	8.14
12	54.59	59.32	8.25
16	45.00	73.81	8.14
20	37.63	87.43	8.26
24	29.44	113.29	8.18

(a) UHM solver

	Total cost: 3264 GFLOP		
Cores	Time (sec)	GFLOP/sec	Memory (GB)
1	454.14	7.19	8.328
2	278.40	11.73	10.10
4	166.78	19.58	10.11
8	104.19	31.36	12.16
12	71.89	45.45	12.72
16	60.07	54.39	12.86
20	50.64	64.52	12.61
24	43.83	74.83	12.79

(b) MUMPS

	Total cost: 7490 GFLOP		
Cores	Time (sec)	GFLOP/sec	Memory (GB)
1	1153.07	6.50	9.57
2	610.16	12.28	9.59
4	309.32	24.22	9.65
8	173.55	43.16	9.68
12	119.79	62.53	9.77
16	92.38	81.08	9.83
20	77.15	97.09	9.89
24	66.28	113.53	9.89

(c) PARDISO

Table 5.3: Performance benchmark of different direct solvers with a varying number of cores. The test problem has 357,889 DOFs and discretized with $p = 4$.

	Total GFLOP	GFLOP/sec	Time (sec)	Memory (GB)
UHM	**3476**	**113.29**	**29.44**	**8.18**
MUMPS	**3264**	74.83	43.83	12.79
PARDISO	7490	**113.53**	66.28	9.89

Table 5.4: Performance summary of the sparse direct solvers for 24 cores.

Order p	# of DOFs	# of non-zeros
1	6,017	524,288
2	45,825	3,276,800
3	152,193	13,107,200
4	357,889	40,140,800
5	695,681	102,760,448
6	1,198,337	231,211,008

Table 5.5: Sparse matrices are generated based on a unstructured tetrahedral mesh corresponding to a unit spherical domain with a varying order of approximation from 1 to 6.

- With the controlled computational cost and high performance floating point operations, the UHM solver with 24 cores achieves respectively 1.48x and 2.25x speed-up compared to MUMPS and PARDISO.

Figure 5.6 compares the strong scaling of the solvers and their memory usages as a function of the number of cores. Our solver achieves 16x speed-up on 24 cores while MUMPS and PARDISO respectively achieve 11x and 7x speed-up. The graph also shows that the memory usage in our solver does not grow with the number of cores being used.

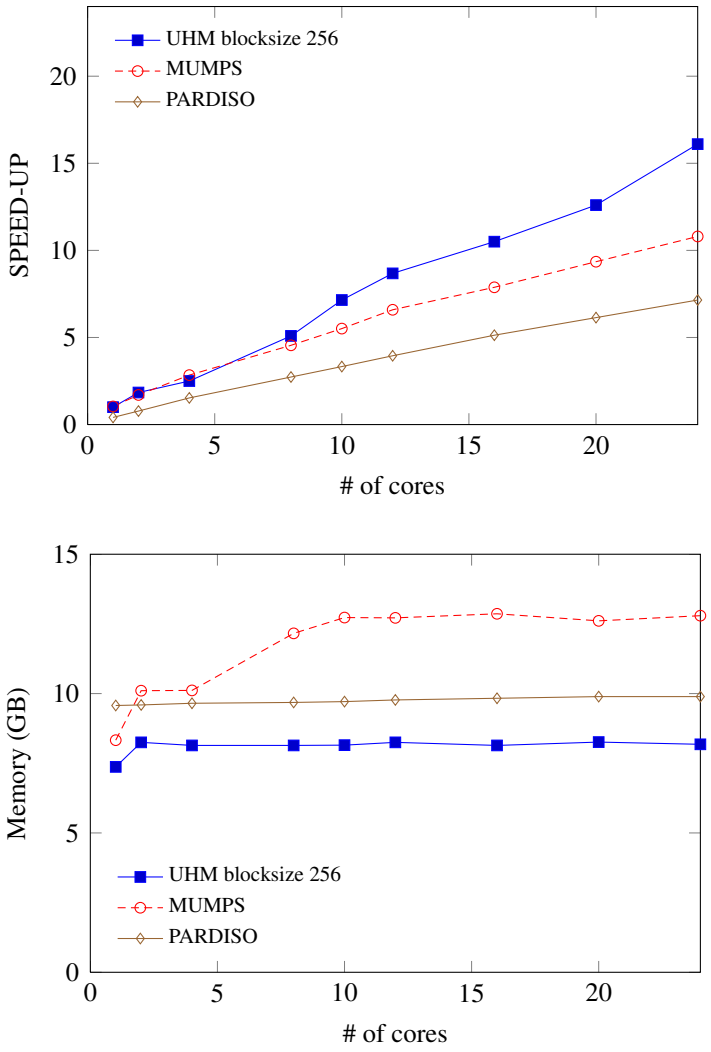


Figure 5.6: Performance in the factorization phase for a fixed problem discretized with $p = 4$. A reference of the speed-up graph is a sequential performance of the UHM solver.

Analysis phase

Our test problems are based on the same tetrahedral mesh with a varying polynomial order of approximation p from 1 to 6. The problem sizes and sparsity

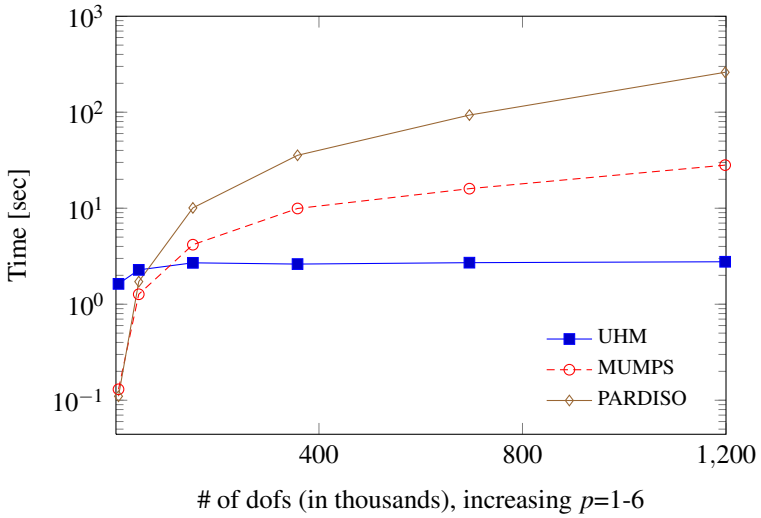


Figure 5.7: Time (lower is better) measured in the analysis phase.

are described in Table 5.5.

Figure 5.7 shows that our solver is highly efficient when a sparse system is based on a high order discretization. The higher efficiency of our solver can be attributed to its unique interface that takes into account the *hp*-discretization process. Instead of partitioning the matrix graph itself, we construct a weighted quotient graph based on the element connectivity, where the weights are associated with nodal Degrees of Freedom (DOFs). Hence, the time complexity does not vary if we only change the polynomial orders. On the other hand, MUMPS and PARDISO spend a considerable amount of time in reordering and analyzing the matrix that increases with higher p.

p	UHM	MUMPS
1	1,104	658
2	9,527	3,613
3	32,149	8,829
4	**62,955**	18,646
5	**63,033**	30,583
6	**63,042**	**65,400**

# of fronts

p	UHM	MUMPS
1	694	594
2	2,508	2,661
3	5,746	6,025
4	10,197	10,343
5	16,171	16,541
6	22,336	23,380

Max front size

Table 5.6: Frontal matrices used in the factorization.

Table 5.6 compares the frontal matrices constructed during factorization [1]. An interesting observation is that two different solvers report similar maximal front sizes. This implies that the asymptotic FLOPs and required memory would be almost the same for the two solvers. However, the number of fronts is distinctly different for different p.

For the example of $p \geq 4$, the hp-mesh populates DOFs for all topological nodes *i.e.*, vertices, edges, faces and interior nodes. As we analyze a quotient graph based on the hp-mesh, our solver reports a similar number of frontal matrices when $p \geq 4$. In contrast, MUMPS does not recognize this topological structure.

Numerical factorization phase

Figure 5.8 shows that our solver is more efficient than others for problems derived with higher p. MUMPS and the UHM solver report roughly the same num-

[1] For this comparison, we do not evaluate PARDISO as it is implemented with left-looking supernodal factorization.

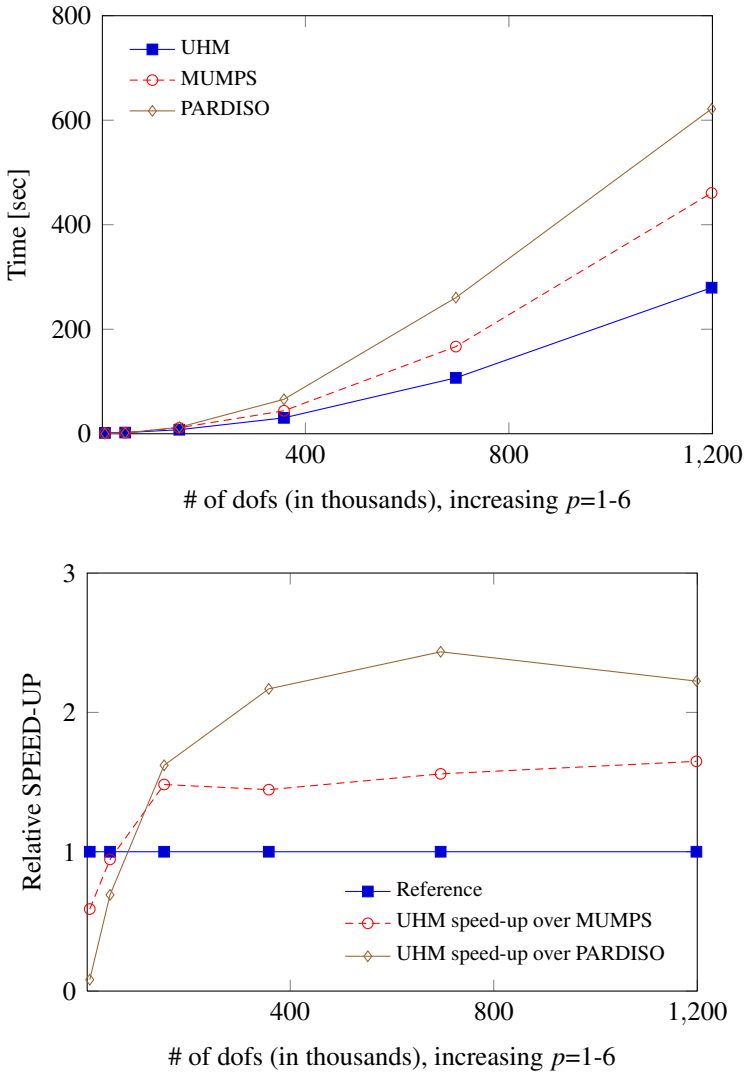


Figure 5.8: Time complexity in the factorization phase (24 cores are used).

ber of FLOPs estimates for the factorization. Hence, the performance gain over MUMPS is largely due to the efficient use of level 3 BLAS operations and asynchronous parallel execution of fine-grained tasks. On the other hand, MUMPS also suffers from more communication overhead created by explicitly managed MPI.

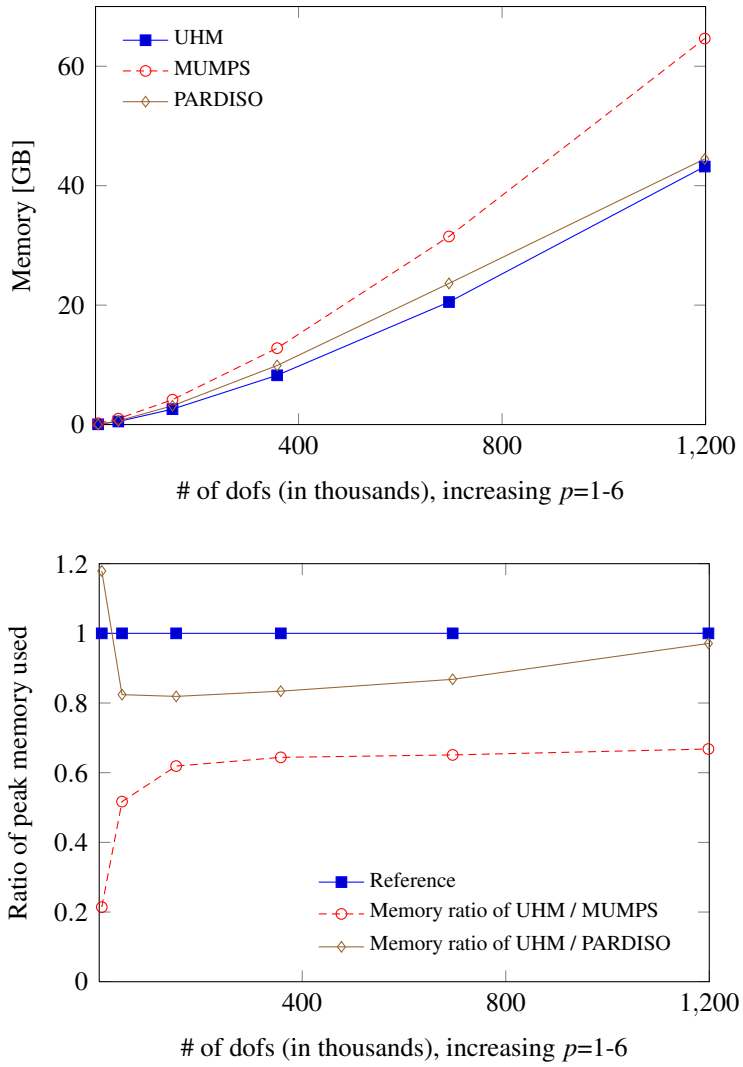


Figure 5.9: Peak memory used in the factorization phase (24 cores are used).

While the PARDISO solver outperforms the others for the problem with the lowest order approximations, the solver performs slower for problems with higher p. The number of FLOPs in the factorization phase is twice as large as the others. We were not able to determine why PARDISO reports a different computational

cost compared to others.

Figure 5.9 compares the space complexity of the solvers: the graph compares the peak memory used for solving the problems. Our solver shows substantial memory saving compared to other solvers. Compared to MUMPS, our solver uses 30% less memory for high p. We also see that our solver uses 20% less memory than the PARDISO solver. This is probably due to the fact that other sparse direct solvers over-allocate the workspace to accommodate the delayed pivots and additional fills. On the other hand, our solver dynamically creates or destroy matrices when UHMs are assembled during the factorization.

5.3 Scheduling tasks to multiple GPUs

When a heterogeneous architecture with multiple Graphic Processing Units (GPUs) is used for large-scale sparse matrix factorization, a naive task offloading is unlikely to be efficient due to the irregular nature of the density in the sparse matrix. In the course of multifrontal factorization, the sizes of encountered subproblems vary significantly, and these irregularly grained problems are the main concern for performance:

- at the start, there exist many small-sized subproblems, and those problems are mostly not large enough to yield performance gains from GPU offloading;
- on the other hand, the large subproblems at the upper-level hierarchy may not fit in the local storage of a GPU.

Our factorization scheme using algorithms-by-blocks is still viable to solve these issues. In the hybrid context, where both a CPU and a GPU are utilized, a problem with this approach is that a single algorithmic block size is not apt to both GPUs and the host processor. For instance:

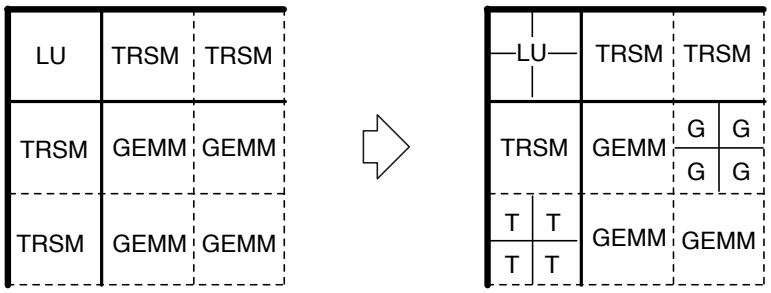
- tasks should be coarse enough to obtain performance from GPUs over the invested overhead such as data transfers;
- fine-grained tasks are preferred for the multi-core processor to improve parallel efficiency and workload balance in cores.

This task granularity issue is more significant when the computational power of GPUs and the host multi-core processor is greatly imbalanced.

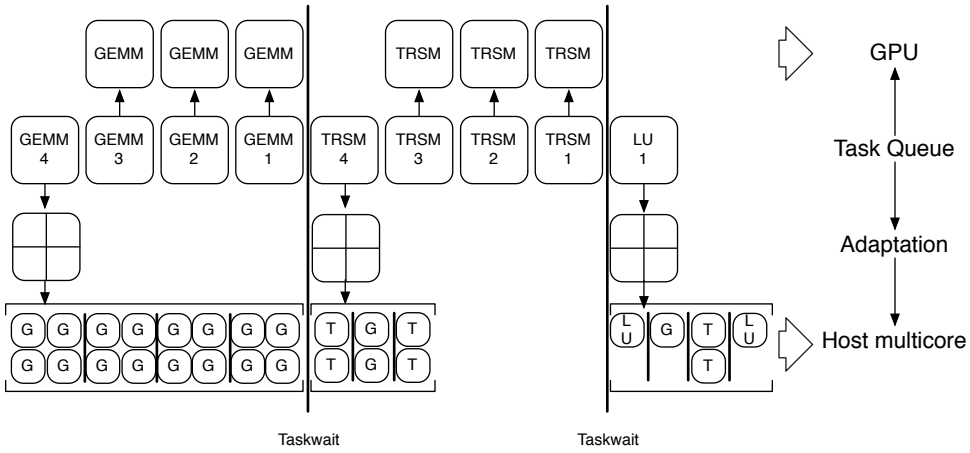
To overcome these conflicting performance issues, nonuniform block sizes are used in our solver. Initially, element matrices are partitioned with a block size that is best-suited for GPUs. Coarse grain tasks are first generated using algorithms-by-blocks, and they are scheduled to GPUs. Simultaneously, some of those tasks are further refined with a smaller block size to assign them to the host multi-core processor.

Here we explain this multi-level task subdivision scheme with an example depicted in Fig. 5.10. For simplicity, we consider a target compute node consisting of a multi-core processor and a GPU with following characteristic parameters:

- there is one GPU, hosted on a quad-core CPU;
- the GPU has double the optimal block size b of the CPU; and



(a) Multi-level matrix blocking



(b) Heterogeneous task scheduling

Figure 5.10: Task adaptation and workload balancing on a heterogeneous architecture accelerated by a GPU.

- the GPU is three times faster than the quad-core CPU

We also assume that the GPU is only used for BLAS functions [2].

First, the element matrix is partitioned with the GPU block size $2b$. By applying an algorithm-by-blocks LU factorization to the partitioned matrix, a list of

[2]This restriction of earlier GPUs may no longer hold, but it illustrates the principle that in heterogeneous setups certain operations may only be executable on one type of device.

Lonestar	
Processors	2x Hex-Core, Westmere 3.33 GHz
Memory	24 GBytes
BLAS	Intel MKL 10.3
DLA library	`libflame` ver 6192, CUBLAS
DGEMM Peak	13.3 GFLOPS/core
Interconnection BUS	PCI Express 2.0
GPUs	2x NVIDIA Tesla, M2070 (6 GB RAM)
Compiler	GNU 4.4.5
DGEMM	300 GFLOPS/device

Table 5.7: Specifications of the Lonestar GPU node at TACC.

coarse grained tasks is produced. Considering the three-to-one performance balance of the two devices, we immediately see that one out of every four tasks should be computed on the multi-core processor to achieve global load balance. When a coarse grain task is dispatched to the multi-core processor, the task is dynamically adapted with its block size b. Resulting fine-grained tasks are redistributed to the available cores. For more detailed descriptions of this approach, we refer [48, 49].

5.4 Performance on a heterogeneous architecture with multiple GPUs

In this section, we demonstrate the performance of our sparse direct solver on a heterogeneous architecture. Experimental results are again obtained from a single GPU node on *Lonestar*: 12x multi-core processor with two GPUs. Detailed specification of this machine is tabulated in Table 5.7 [3]. For performance evaluation, we use the same problem as one used in Section 5.2 with a uniform order of

[3]Note that this machine is different from the large memory node which is used in Section 5.2.

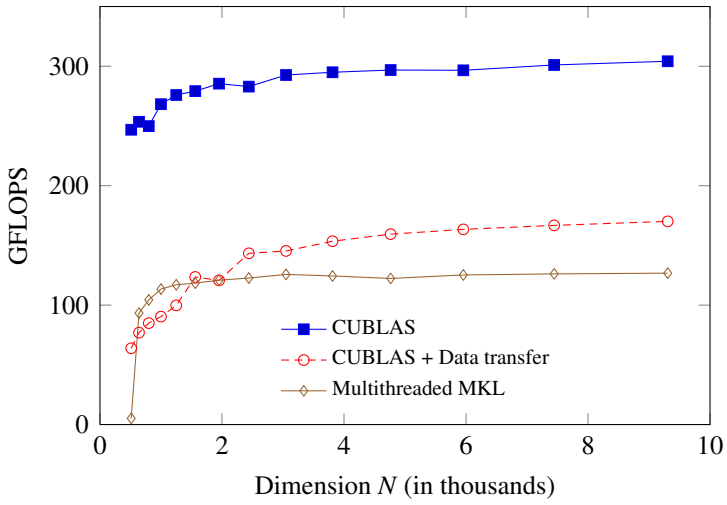


Figure 5.11: Dense DGEMM performance with M = N and K = 1024 for a 12x Intel Xeon X5680 @ 3.33GHz processor and a single Fermi GPU M2070.

approximation, $p = 4$.

Node characterization

The machine is equipped with two *Fermi* GPUs, where each GPU delivers a peak performance of 300 GFLOPS for DGEMM, whereas the multi-core processor delivers 13 GFLOPS per a core performing DGEMM.

Extracting the most performance from different devices, we use only two performance parameters (*i.e.*, block size and performance ratio) at the setup phase. However, several factors such as individual performance profiles, different memory hierarchy, data transfers (or caching policies) interact in determination of these two parameters, and some of them are not clearly identified *a priori*. In this benchmark, the following parameters are used;

- there is one 12-core CPU with a block size of 256,
- there are two GPUs with a block size of 2048, and
- a single GPU is 3 times faster than the full CPU.

In principle, block sizes for different types of devices can be empirically determined by running individual test problems like one depicted in Fig. 5.11. If we include the data transfer cost in DGEMM performance, the GPU with a block size below 2000 does not provide any performance over the host processor in our computing model. This performance graph also shows that an order of magnitude higher device performance can be achieved if the data on GPU memory are managed/reused.

Scalability

In Fig. 5.12, we compare the dense factorization accelerated by multiple GPUs. Little performance is obtained from GPUs for the small sized problems, but the asymptotic speed-up of 3.5x from two GPUs is observed for large problems. As the most computational complexity comes from the large element matrices, the higher asymptotic performance offers an opportunity to use GPUs more efficiently for large scale multifrontal factorization.

Table 5.8 describes the performance of our sparse solver varying the number of GPUs used. For this model sparse problem, the asymptotic speed-up for dense problems (3.5x) seen in Fig. 5.12 is not observed due to the range of subproblem sizes and irregularity of the sparse problem. This model problem creates dense

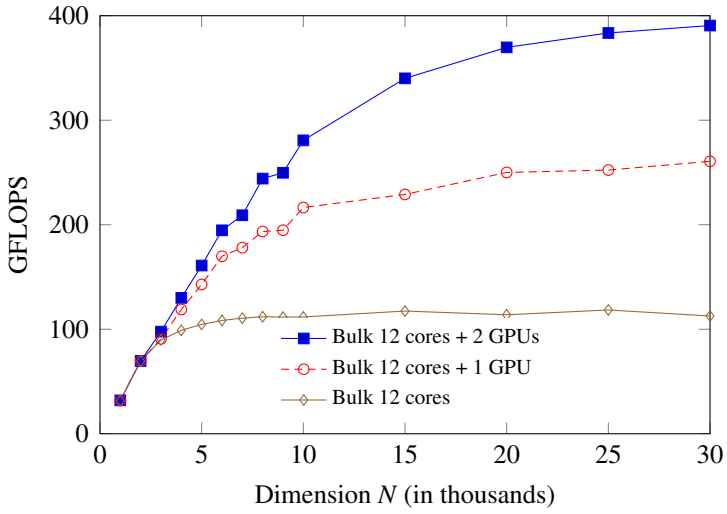


Figure 5.12: Dense LU factorization without pivoting accelerated by multiple GPUs.

		Performance		Speed-up	
Cores	GPUs	Time [sec]	GFLOP/sec	vs 1 core	vs 12 cores
1	0	291	11.13	1.00	-
2	0	213	15.21	1.36	-
4	0	85	38.11	3.42	-
8	0	45	71.98	6.46	-
12	0	33	98.16	8.81	1.00
12	1	23	140.84	12.65	1.43
12	2	19	170.50	15.31	1.69

Table 5.8: Sparse LU with partial pivoting accelerated by multiple GPUs. Note that the sparse system has 357,889 DOFs and a max frontal matrix size is 10,791.

submatrices upto a size 11k, which is expected to lead at most 2.51x speed-up as depicted in Fig. 5.12. However, being aided by two GPUs, we are able to obtain 1.69x speed-up in this example problem.

5.5 Summary

In this chapter, we presented a two-level task scheduling approach for the multifrontal factorization of a matrix. First-level tasks are generated via the post-order tree traversal; next, those tasks are decomposed into fine-grained tasks using algorithms-by-blocks. By constructing a DAG of the fine-grained tasks for an encountered dense subproblem, those tasks are asynchronously executed while respecting their dependencies. As task schedulers are invoked following the partial order of elements in the assembly tree, we achieve efficient task scheduling with low overhead, without the need for the construction of a global DAG.

From the perspective of implementation, our task scheduler is particularly designed for multiple dense problems. This feature is necessary for multifrontal factorization as the method converts sparse matrix problem into a set of dense subproblems. Our task scheduler exploits the nested parallelism supported by OpenMP tasking. This enables nested multi-level DAG scheduling. We demonstrated the efficiency of our approach by comparing with state-of-the-art sparse direct solvers. Our solver outperforms the others when a system of equations is based on higher order discretization.

Responding to emerging heterogeneous GPU architectures, we introduced a simple but efficient task subdivision scheme. Our approach dynamically adapts task granularity by adjusting a block size to be suitable for the device that is used. For instance, a large block size is selected for the efficient use of GPUs while a smaller block size is used to improve concurrency in a multi-core processor. The main benefit of this approach is in portable performance. Two tuning parameters are

used to characterize various kinds of heterogeneous architectures. We also showed that our approach is viable by applying the technique to both dense and sparse problems.

Chapter 6

Numerical Results

In this chapter, we present numerical examples that have been conducted to verify our FE code and results for the coupled BioHeat Transfer (BHT) problem. Overall, we will follow a verification procedure similar to the one used in [35, 97]. This procedure includes the following numerical experiments:

- evaluation of geometry error and verification of mesh generation;
- verification of the Electro-Magnetic (EM) code against manufactured solutions;
- verification of the Perfectly Matched Layer (PML) by solving EM wave scattering problems.

After the EM code is thus verified, we present a few case studies for the BHT problem coupled with Maxwell equations.

6.1 Geometry error and verification of the curvilinear mesh generation

A precise description of geometry is critical for high order FEs. As mentioned earlier, our computational domain is partitioned into Geometry Modeling

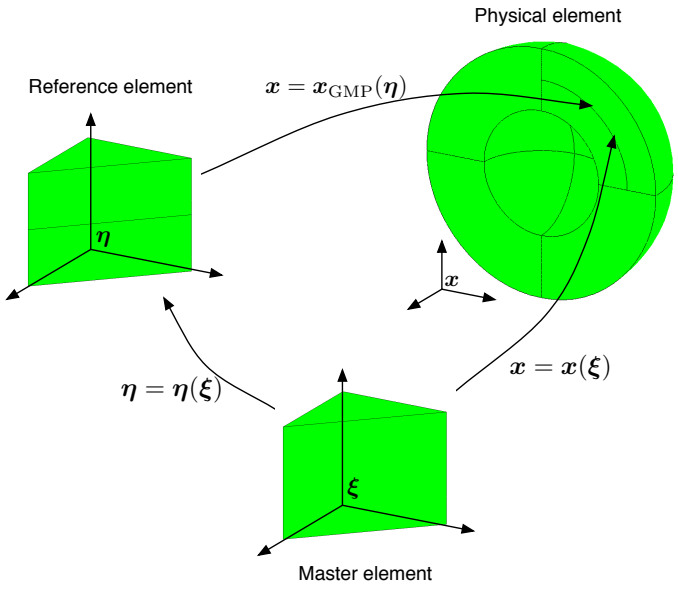


Figure 6.1: Element mapping. Coordinates of $\boldsymbol{\xi}$, $\boldsymbol{\eta}$, and $\mathbf{x}$ represent master, reference, and physical element coordinates respectively.

Package (GMP) blocks that coincide with an initial FE mesh. For each block, GMP provides a parameterization that maps a corresponding reference GMP block onto a physical one. These parameterizations are globally C^0-conforming; mesh refinements are performed in the reference GMP blocks.

The *exact geometry* element map is constructed by composing an affine map from the master element onto the corresponding element in the reference domain, with the exact GMP parameterization. With this geometry map, the geometry is exactly represented and no geometry error is present.

In practice, the exact geometry element is replaced by the *isoparametric element* with a polynomial approximation constructed using H^1-conforming element shape functions. This approximation is performed through the Projection-Based In-

terpolation (PBI) of the exact GMP parameterizations in the reference domain [27]. The PBI is invariant under a similarity map [1] but not invariant under a general affine transformation. Unfortunately, in the process of refining tetrahedral elements in the reference domain, we obtain elements that are not similar to the master tetrahedral element. Consequently, the PBI of the element map is not longer equivalent with the PBI done in the reference domain. This does not happen for hexahedral and prismatic elements.

As the interpolation of the GMP parameterizations is done in the reference domain, the corresponding geometry error is defined using the H^1 norm in the reference domain. More precisely, if

$$\mathbf{x}_{\text{GMP}} : \Omega_b \ni \boldsymbol{\eta} \rightarrow \mathbf{x} \in \Omega$$

denotes the GMP parameterization for a particular reference block Ω_b, the corresponding geometry error is defined as

$$\text{err}_x = \frac{\left\|\mathbf{x}_{\text{GMP}} - \mathbf{x}_{hp}\right\|_{H^1,\Omega_b}}{\left\|\mathbf{x}_{\text{GMP}}\right\|_{H^1,\Omega_b}},$$

where $\mathbf{x}_{hp}$ denotes the interpolant computed by PBI over the mesh in the reference domain. We verify the generation of geometry Degrees of Freedom (DOFs) through PBI by studying the global h-convergence rates of the PBI error. The theoretical convergence rates are of the form

$$\text{error} \approx Ch^{\min\{p,r\}},$$

[1] Rigid body motion composed with a simple (isotropic) scaling.

where h is the element size, C denotes a generic constant, p is the uniform element order and r is the regularity of the GMP parameterizations measured in Sobolev norms. With the number of DOFs proportional to p^3/h^3, the convergence rate in terms of the total number of DOFs N (with fixed p) is

$$\text{error} \approx CN^{-\min\{p,r\}/3}.$$

Reproduction of a plane wave. As our first example, we solve a Neumann boundary value problem for the time-harmonic Maxwell equations in a spherical domain:

$$\begin{aligned} \nabla \times \left(\tfrac{1}{\mu_r}\nabla \times \mathbf{E}\right) - \omega^2 (\varepsilon_r - j\varepsilon_\sigma)\mathbf{E} &= 0 && \text{in} \quad \Omega, \\ \mathbf{n} \times \left(\tfrac{1}{\mu_r}\nabla \times \mathbf{E}\right) &= -j\omega \mathbf{J}_s^{imp} && \text{on} \quad \Gamma_N, \end{aligned}$$

where Ω represents a spherical domain with a diameter $d = 2\lambda$ and λ represents the wavelength in the medium. For verification purposes, the spherical domain is discretized using both tetrahedral elements and prismatic elements as shown in Fig. 6.2. A Neumann boundary condition is imposed on the surface of the prismatic layer and the boundary data is generated from a manufactured plane wave solution:

$$\mathbf{E}(\mathbf{x}) = \mathbf{E}_0 e^{-j\mathbf{k}\cdot\mathbf{x}}.$$

Here, the polarization vector $\mathbf{E}_0$ and propagating wave vector $\mathbf{k}$ must satisfy the following condition:

$$\mathbf{E}_0 \cdot \mathbf{k} = 0.$$

For convenience, we set them as $\mathbf{E}_0 = (1,0,0)$ and $\mathbf{k} = (0,0,\pi)$. The convergence

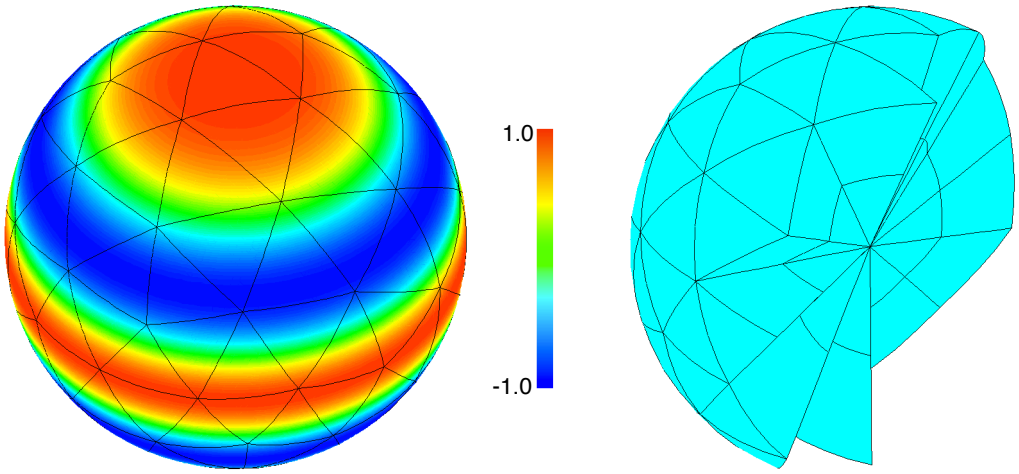


Figure 6.2: E_x component of the plane wave solution.

rates for the solution $\mathbf{E}$ to the Maxwell problem are measured in the $\mathbf{H}(\mathrm{curl})$ norm defined in the physical domain.

$$\mathrm{err}_{\mathbf{E}} = \frac{\|\mathbf{E}-\mathbf{E}_{hp}\|_{\mathbf{H}(\mathrm{curl}),\Omega}}{\|\mathbf{E}\|_{\mathbf{H}(\mathrm{curl}),\Omega}}$$

With tetrahedral and prismatic elements of order $(p+1/2)$ [2], the expected rates of convergence are identical with those for the geometry error.

A detailed convergence history for both geometry and solution errors is tabulated in Table 6.1. Note that the problem size corresponding to $p=3$ and $p=4$ meshes after the third global mesh refinement is larger than the limited memory space (100 GB) of our workstation. The obtained convergence rates coincide with ones predicted by the theory and indicate that the geometry maps are at least of H^5

[2] Nédélec's elements of the first type.

		Geometry		Solution	
	# of refine.	err_G	rate	err_E	rate
$p=2$	0	0.55047E-02	n/a	0.25671E-01	n/a
	1	0.14172E-02	-0.689	0.65863E-02	-0.675
	2	0.35724E-03	-0.679	0.16654E-02	-0.671
	3	0.89576E-04	-0.673	0.42027E-03	-0.667
$p=3$	0	0.28910E-03	n/a	0.27565E-02	n/a
	1	0.37623E-04	-1.015	0.35990E-03	-1.000
	2	0.47491E-05	-1.012	0.45640E-04	-1.003
	3	0.59512E-06	-1.007	-	-
$p=4$	0	0.12783E-04	n/a	0.24933E-03	n/a
	1	0.83480E-06	-1.345	0.16572E-04	-1.324
	2	0.52771E-07	-1.344	0.10557E-05	-1.334
	3	0.33109E-08	-1.339	-	-

Table 6.1: Convergence rates of both the geometry and the solution on a series of globally refined meshes.

regularity.

This sanity check does not only verify the interpolation routines but also provides important information about the regularity of the GMP parameterizations that affects the overall convergence rates for the FE solution.

6.2 Electro-Magnetic wave scattering problems

For a general EM scattering problem, we consider an incident EM wave $(\mathbf{E}^{\text{inc}}, \mathbf{H}^{\text{inc}})$ satisfying Maxwell equations in the entire domain $\Omega \subset \mathbb{R}^3$. We separate the interior domain of a scatterer represented by $\Omega_{\text{int}} = \bigcup_{i=1}^{n} \Omega_i$, where Ω_i denotes a subdomain occupied by the object. This geometry is illustrated in Fig. 6.3, where

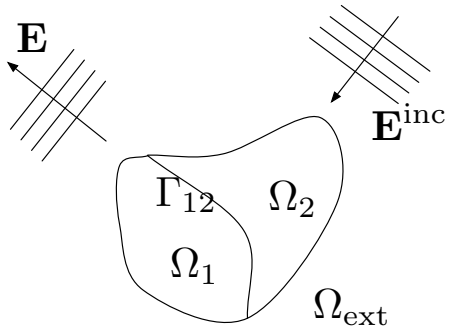


Figure 6.3: A scattering object places in the domain of interest (Ω) with an incident wave $\mathbf{E}^{\text{inc}}$.

Ω_{ext} denotes the exterior domain outside the scatterer and Γ_{ij} represent the interface between different materials.

Let us begin with the non-dimensionalized second-order wave equation discussed earlier; that is

$$\nabla \times \left(\frac{1}{\mu_r} \nabla \times \mathbf{E} \right) - \omega^2 \left(\varepsilon_r - j\varepsilon_\sigma \right) \mathbf{E} = -j\omega \mathbf{J}^{\text{imp}}. \tag{6.1}$$

We assume that the materials are isotropic and homogeneous in each subdomain Ω_i. However, the material properties are not necessarily the same among different subdomains. Next, we decompose the total electric field into the scattered wave $\mathbf{E}$ and the incident wave $\mathbf{E}^{\text{inc}}$:

$$\mathbf{E}^{\text{tot}} = \mathbf{E} + \mathbf{E}^{\text{inc}}. \tag{6.2}$$

Then, the total electric field should satisfy the Maxwell equations subject to boundary conditions on the surface $\partial\Omega$. In addition, the incident wave satisfies the reduced wave equation in the free space:

$$\nabla \times \nabla \times \mathbf{E}^{\text{inc}} - \omega^2 \mathbf{E}^{\text{inc}} = 0,$$

where ω corresponds to the free space wave number. Assuming no current source inside of the scatterer, we obtain the following equation by substituting (6.2) into (6.1):

$$\begin{aligned}\nabla\times\left(\frac{1}{\mu_r}\nabla\times\mathbf{E}\right)-\omega^2\hat{\varepsilon}_r\mathbf{E} &= -\nabla\times\left(\frac{1}{\mu_r}\nabla\times\mathbf{E}^{\text{inc}}\right)+\omega^2\hat{\varepsilon}_r\mathbf{E}^{\text{inc}}\\ &= -\frac{1}{\mu_r}\left(\underbrace{\nabla\times\nabla\times\mathbf{E}^{\text{inc}}-\omega^2\mathbf{E}^{\text{inc}}}_{0}-\omega^2\left(\hat{\varepsilon}_r\mu_r-1\right)\mathbf{E}^{\text{inc}}\right)\\ &= \omega^2\left(\hat{\varepsilon}_r-\mu_r^{-1}\right)\mathbf{E}^{\text{inc}},\end{aligned}$$

where $\hat{\varepsilon}_r=\varepsilon_r-j\varepsilon_\sigma$ is defined as a complex relative permittivity. By defining the equivalent volume current $\mathbf{J}^{\text{imp}}$ in the scatterer

$$-j\omega\mathbf{J}^{\text{imp}}=\omega^2\left(\hat{\varepsilon}-\mu_r^{-1}\right)\mathbf{E}^{\text{inc}}, \tag{6.3}$$

the homogeneous wave equation for the total electric field is transformed to a non-homogeneous wave equation for the scattered field. This formulation changes the $\mathbf{E}^{\text{inc}}$ to the equivalent electric volume current $\mathbf{J}^{\text{imp}}$.

To complete this formulation, proper boundary conditions should be imposed. For a Perfect Electric Conductor (PEC) surface, the total electric field $\mathbf{E}$ should vanish and this leads a non-homogeneous Dirichlet condition for the scattered field:

$$\mathbf{n}\times\left(\mathbf{E}+\mathbf{E}^{\text{inc}}\right),\ \text{on}\ \partial\Omega.$$

In addition, we impose the Silver-Müller radiation condition at infinity to ensure that the solution is uniquely determined. Since the FE mesh cannot be constructed for an unbounded domain, the infinite domain should be properly truncated with an

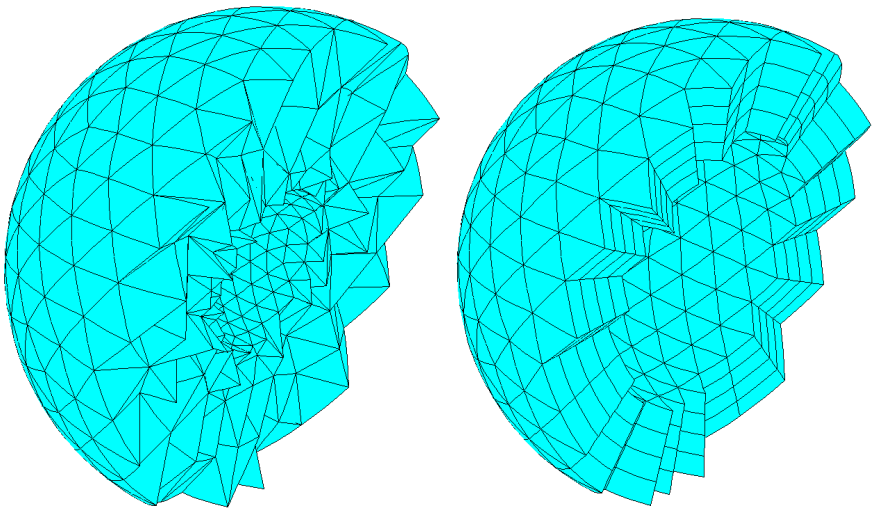
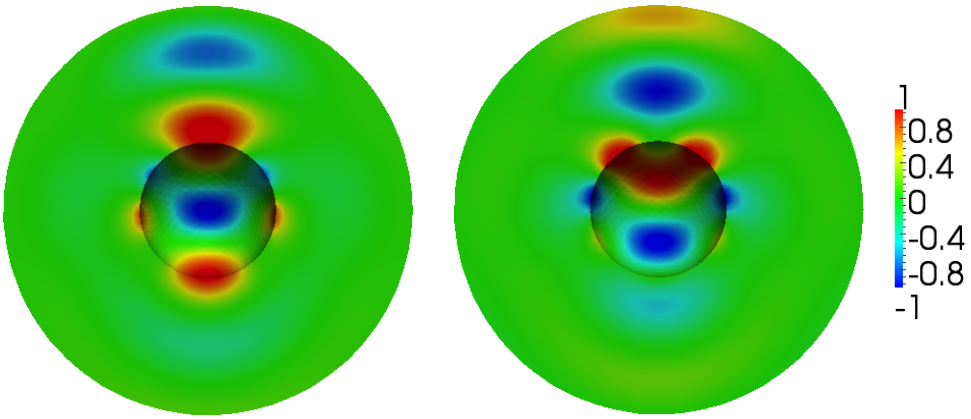


Figure 6.4: A cutaway view of the spherical shell for the PEC scattering problem. Note that PML is rescaled for the visualization purpose.

absorbing boundary condition mimicking the outgoing waves. As discussed earlier, we use the PML technique and follow the construction in [60].

Next, we verify the EM code by solving EM scattering problems: a plane wave scattering on a PEC and a dielectric sphere. The numerical solution will be compared against the analytic Mie series solution; the exact solution was implemented using the algorithms described by Wiscombe [95].

Scattering of a plane wave on a PEC sphere. In this example, we set up a unit-radius sphere (non-dimensional) with an incident plane wave of which wavelength is λ. Figure 6.4 shows the cutaway view of the problem domain. The free space domain ($1 \leq r \leq 3$) is composed of tetrahedral elements. The PML domain is



Real part of E_x Imaginary part of E_x

Figure 6.5: A slice view normal to y-axis of the manufactured wave solution for the PEC scattering problem.

extruded with prismatic elements of a thickness $t = \lambda$; the layer is anisotropically refined and we use four elements to the thickness direction. Figure 6.5 describes a manufactured solution for a case when the diameter of the scatterer coincides with the free space wavelength λ. The actual frequency is

$$\text{frequency} = \frac{\omega}{2\pi a\sqrt{\varepsilon_0\mu_0}},$$

where a is a reference length. By relating with a scale of the human head $a = 0.1m$, the frequency used in this example corresponds to 1.5 GHz. In Table 6.2, we report the relative $\mathbf{H}(\text{curl})$ error against the Mie series solution for different wavelengths. The use of higher order elements give us an accurate and cost-effective solution. We also observe that the error is only slightly reduced when the approximation order increases from $p = 3$ to $p = 4$. This is probably because our PML is not precisely tuned; a discrete version of the PML may not resolve the rapidly changing

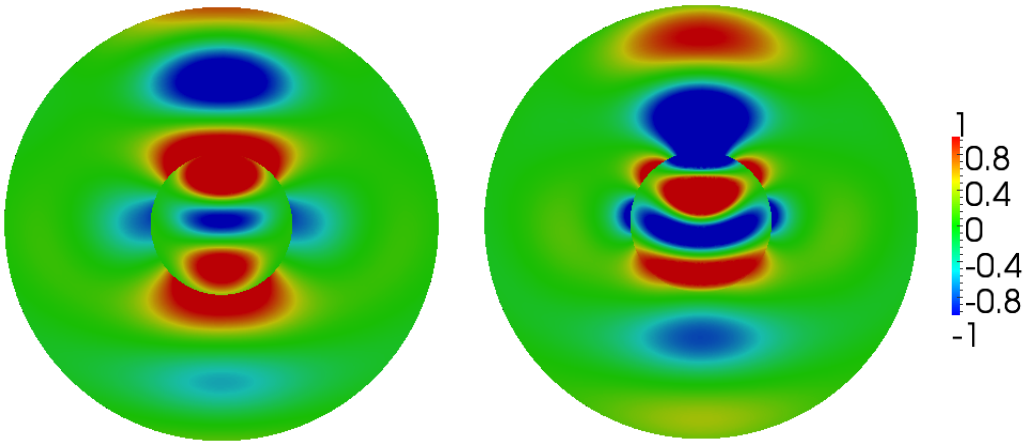
	λ/h		
p	8	4	2
1	0.25184E+00	0.53327E+00	0.10921E+01
2	0.58567E-01	0.11029E+00	0.64884E+00
3	0.52356E-01	0.72234E-01	0.23300E+00
4	0.52017E-01	0.70954E-01	0.20881E+00

Table 6.2: PEC scattering. The relative solution error is measured in the standard $\mathbf{H}(\text{curl})$ norm against the manufactured Mie series solution.

solution behavior and this can result in spurious reflections. The optimal PML discretization can be obtained by using hp-adaptivity [62]. However, we have not used an automatic adaptivity in this work and the same PML parameters have been used for all test cases.

Scattering of a plane wave on a dielectric sphere. We use two layered spheres ($r = 1,3$), representing the scatterer and the free space domain. These spheres are modeled with tetrahedral elements. The domain of interest is surrounded by prismatic elements to construct a PML, for which the thickness corresponds to a single free space wavelength. Element sizes are locally adjusted to account for the dielectric medium; four elements per wavelength are used in both the dielectric sphere and the free space including the PML.

The Mie series solution and the EM material properties of the dielectric sphere are illustrated in Fig. 6.6. Numerical results obtained from two domains are tabulated in Table 6.3. An observation is that the numerical error in the scatterer is much smaller than the error in the free space domain. This is probably because



Real part of E_x Imaginary part of E_x

Sphere diameter (=λ)	ε_r	μ_r	σ_r
2.0	4.0	1.0	0.0

Material properties of the dielectric sphere

Figure 6.6: A slice view normal to y-axis of the manufactured wave solution for the dielectric scattering problem. The internal wave (E_x^{tot}) is plotted inside of the scatterer and the scattered wave (E_x) is plotted outside of the sphere.

	Relative error	
p	Scatterer	Free space
1	0.56911E+00	0.53326E+00
2	0.15919E+00	0.14165E+00
3	0.53515E-01	0.11358E+00
4	0.47789E-01	0.11358E+00

Table 6.3: Scattering waves in a dielectric sphere. The relative solution error is measured in the standard $\mathbf{H}(\text{curl})$ norm against the manufactured Mie series solution.

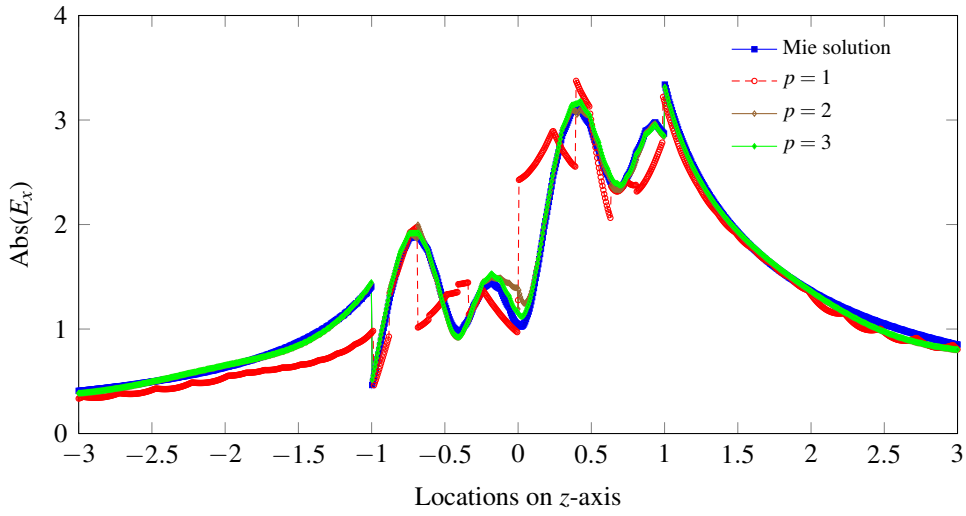


Figure 6.7: The magnitude of the electric field E_x scattered by the dielectric sphere, where the internal field (E_x^{tot}) is plotted for $r \leq 1$.

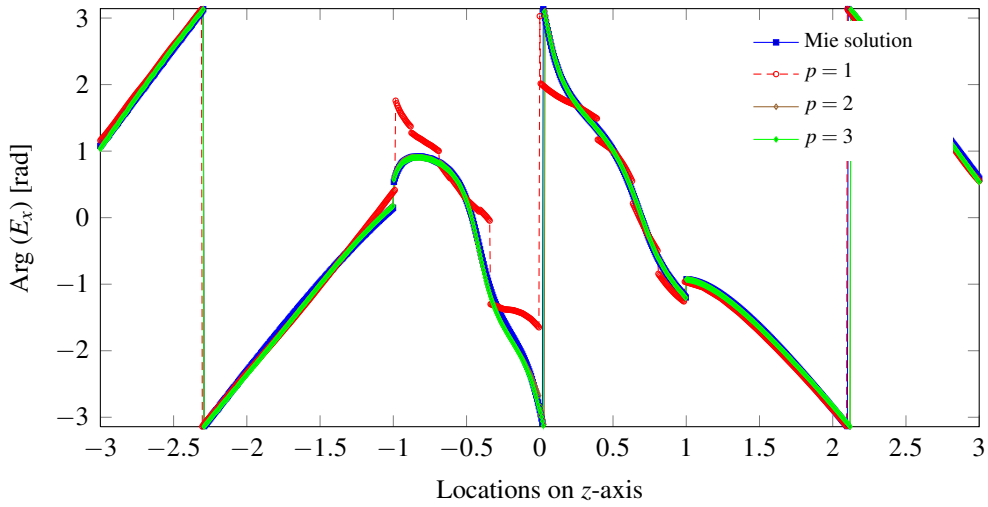


Figure 6.8: The phase angle of the electric field E_x component scattered by the dielectric sphere, where the internal field (E_x^{tot}) is plotted for $r \leq 1$.

the discrete version of the PML, which is not precisely tuned, incurs an irreducible error in the domain of interest. We also compare the E_x component sampled over

the z-axis in Fig. 6.7 and Fig. 6.8. The graph shows that our numerical solution approximated with $p = 2,3$ is well-matched to the Mie series solution. In contrast, the solution with $p = 1$ does not capture the propagating wave inside. Note that the standard criterion for wave scattering and radiation problems for linear elements is to use 10 elements per wavelength.

6.3 Transient BioHeat Transfer problem

As discussed earlier, the two problems are weakly coupled with each other. The Maxwell equations are used to compute the Specific Absorption Rate (SAR), which represents the absorbed power density when a tissue is exposed to the EM field. The temperature distribution in the human head is determined by the Pennes BHT equation taking account of SAR as a localized heating source. Here, we recall the Pennes equation as follows:

$$\rho c \frac{\partial u}{\partial t} = \underbrace{\nabla \cdot k \nabla u}_{\text{diffusion of heat}} + \underbrace{W_b c_b (u_{a0} - u)}_{\text{perfusion rate}} + \underbrace{\dot{q}_m}_{\text{metabolism}} + \underbrace{\dot{q}_{\text{SAR}}}_{\text{EM energy}},$$

where

$k =$ effective thermal conductivity of the tissue $[W/m^\circ\text{C}]$,

$c =$ specific heat of the tissue $[J/kg^\circ\text{C}]$,

$u =$ temperature of veins in the tissue $[^\circ\text{C}]$,

$\rho =$ density of the tissue $[kg/m^3]$.

	Head	Free space	PML
# of elements	8,715	26,491	2,472
Element shape	Tetrahedron	Tetrahedron	Prism
Order (p)	3	3	34
Size(cm)	$(h = 26, w = 16, l = 24)$	$r = 36$	$t = \lambda$
Material	Brain	Air	Air

Table 6.4: The phantom model; λ denotes the wavelength of the incident wave $\mathbf{E}^{\text{inc}}$.

	EM properties		Thermal properties		
$\omega = 0.06\pi$	ε_r	ε_s	α	β	h
Brain	4.5805E+01	1.5309E+01	1.4827E-00	1.0405E+01	2.7313E-01

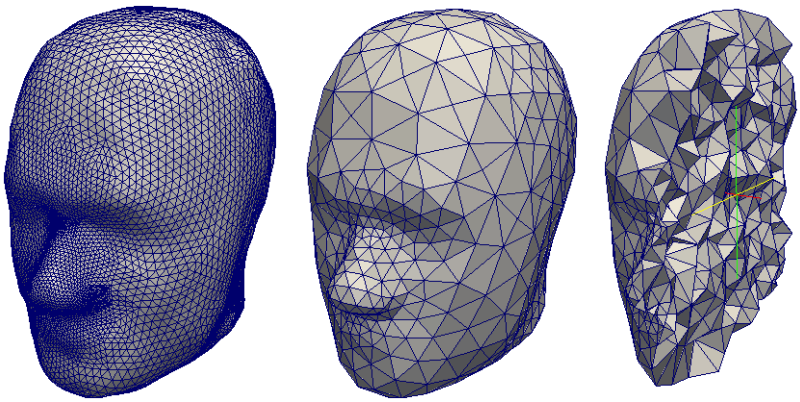
Table 6.5: Non-dimensionalized material parameters used in the phantom model; a reference length $a = 0.01[m]$ and a reference time $t_0 = 1000[sec]$ are used.

We also note that the temperature dependence of the material thermal properties is not considered as the temperature rise due to the deposited EM energy is expected adequately small. However, SAR will be re-evaluated after a certain number of time steps accounting for the temperature dependence of the EM materials. The relation is given in [45]:

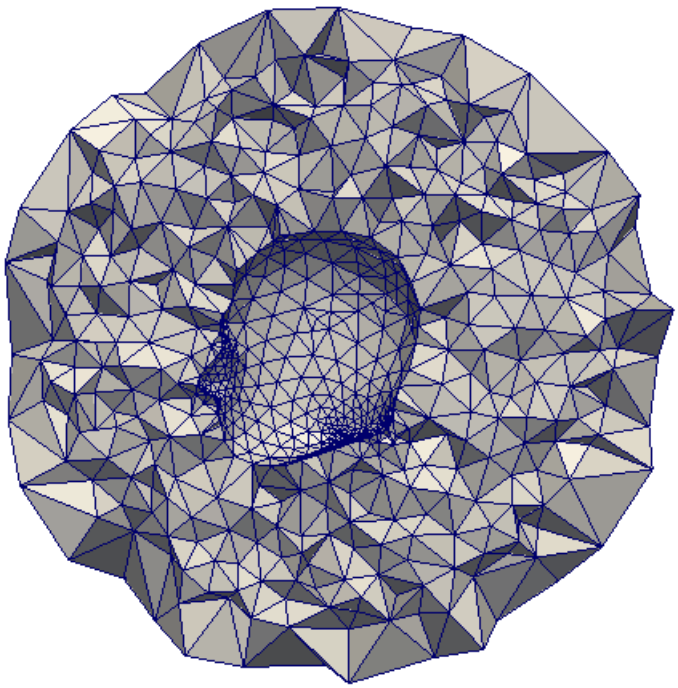
$$\frac{\Delta\sigma}{\sigma} = 0.02\Delta u \text{ and } \frac{\Delta\varepsilon}{\varepsilon} = -0.005\Delta u, \tag{6.4}$$

where σ and ε represent relative conductivity and permittivity, and Δu is temperature difference with a $^\circ$C unit.

Problem set-up. We consider a homogeneous head model, called a *phantom*. The model is obtained from a commercial FE package COMSOL. The surface model is made up with 14,316 triangles. Given this fine-grid surface mesh, GMP reconstructs a smooth G^1-continuous surface [24] that is suitable for the computation



(a) G^1-upgraded mesh (b) Linear mesh (c) Internal view



(d) Free space

Figure 6.9: G^1-continuous phantom model. The figure (a) illustrates the reconstructed curve-linear mesh. The figure (b) and (c) describe the mesh topology that is used in the computation. The figure (d) describes the free space domain.

using higher order elements. A problem in using this precisely defined geometry is the trade-off between the computational efficiency and the quality of representation. We will use isoparametric elements that replace the exact geometry maps in GMP with higher order H^1-conforming shape functions.

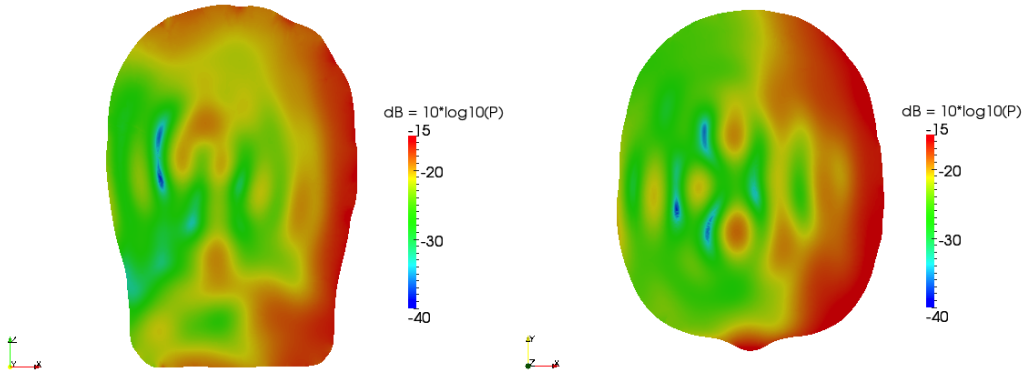
The phantom model represented by the reference GMP blocks coincides with the initial tetrahedral mesh as illustrated in Fig. 6.9; this head model is enclosed by a spherical domain (free space), and the entire domain (including the phantom) is remeshed using TetGen [87]. Finally, the outer surface of the sphere is extruded to construct a PML using prismatic elements. This prismatic layer is anisotropically refined twice along the radial direction; consequently, four elements are used to the thickness direction. A detailed description of the phantom model is tabulated in Table 6.4. Non-dimensionalized electric and material thermal properties used in this study are tabulated in Table 6.5, where they are obtained from [1] and [43] respectively.

Initial Specific Absorption Rate (SAR) distribution. We now compute the SAR distribution in the phantom model. Recall that

$$\dot{q}_{\mathrm{SAR}} = \rho \mathrm{SAR} = \frac{\sigma |\mathbf{E}|^2}{2},$$

where

$$\begin{aligned} \sigma &= \text{conductivity of the tissue} && [(\Omega \cdot m)^{-1}], \\ \mathbf{E} &= \text{electric field intensity} && [V/m], \\ \rho &= \text{density of the tissue} && [kg/m^3]. \end{aligned}$$



Slice view normal to y-axis Slice view normal to z-axis

Figure 6.10: Absorbed power density of the propagating plane wave in the phantom model. This figure is scaled by $dB = 10\log_{10}\left(\sigma|E|^2/2\right)$.

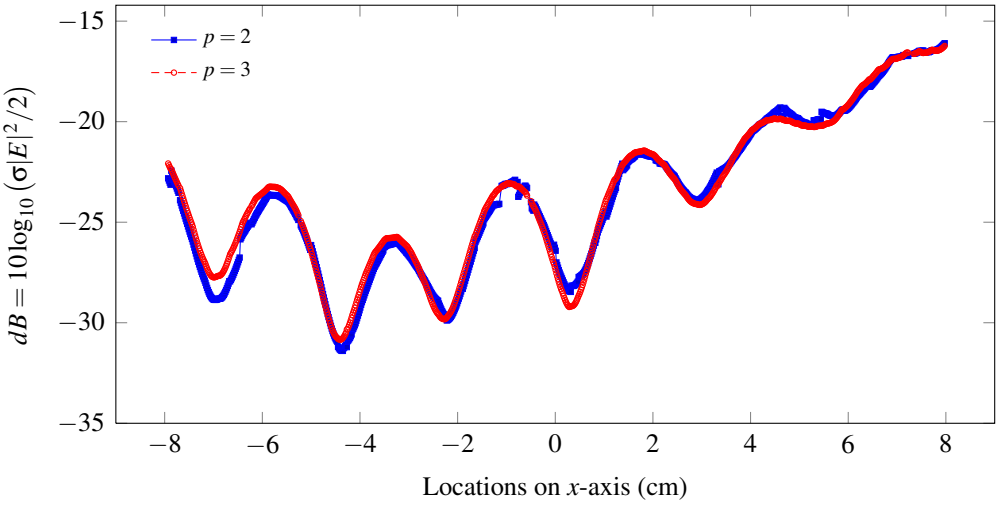
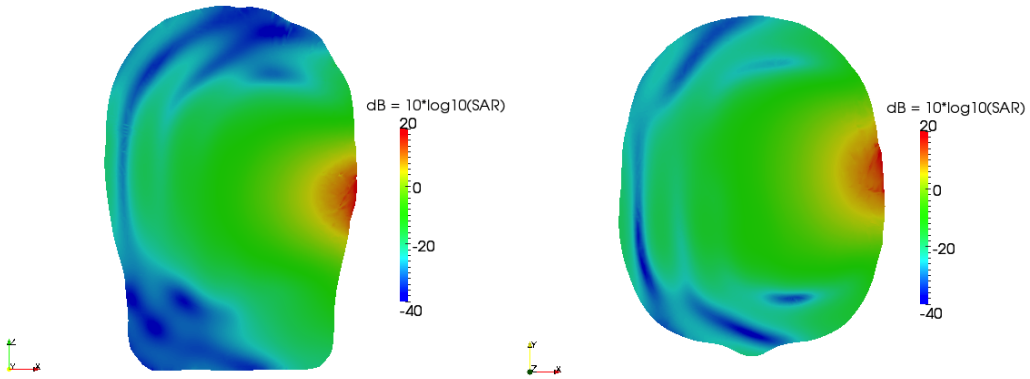


Figure 6.11: Absorbed power density of the propagating plane wave in the phantom model, where the data is sampled over the x-axis.

Two types of incident EM waves are considered in this study: a plane wave and a Hertzian dipole.



Slice view normal to y-axis Slice view normal to z-axis

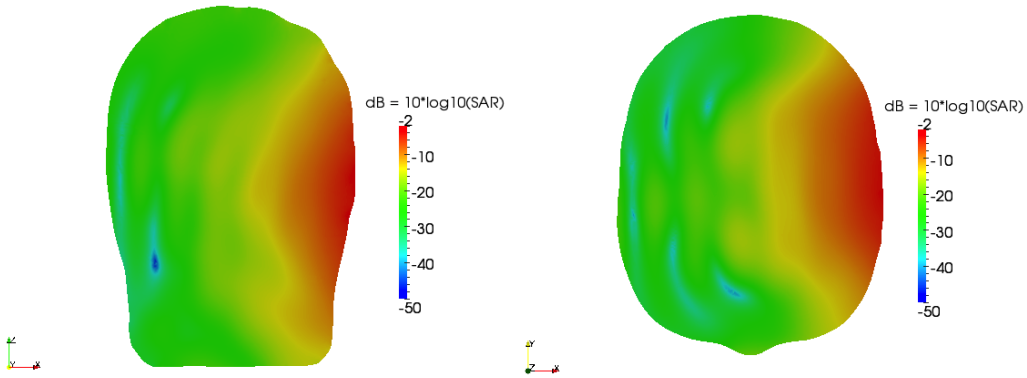
Figure 6.12: SAR distribution in the phantom model exposed to an infinitesimal dipole radiation ($P = 1[W]$) located at $x = 10cm$. This figure is scaled by $dB = 10\log_{10}(\text{SAR})$.

A plane wave is used in this simulation:

$$\mathbf{E}^{\text{inc}} = \mathbf{E}_0 e^{-j\mathbf{k}\cdot\mathbf{x}}, \quad \text{where } E_0 = (0,0,1) \text{ and } \mathbf{k} = \omega(-1,0,0).$$

Figure 6.10 describes the power density of the propagating wave in the phantom model. More quantitatively, we present the power density sampled over the x-axis as depicted in Fig. 6.11. The graph shows the decaying trend of the propagating wave in the lossy medium. We also observe that the solution approximated with $p = 2$ is slightly under-resolved. Regarding to the solution accuracy of our discrete model, we will discuss later in this chapter.

As for our second example, we use a Hertzian dipole to model the EM wave radiation from an antenna of a cell phone. The radiating electric field from the dipole is analytically determined from the magnetic vector potential (see, pages



Slice view normal to y-axis Slice view normal to z-axis

Figure 6.13: SAR distribution in the phantom model exposed to an infinitesimal dipole radiation ($P = 1[W]$) located at $x = 16cm$. This figure is scaled by $dB = 10\log_{10}(\mathrm{SAR})$ and the maximum SAR is found as 0.5950 [W/kg].

55–60 in [44]); the following electric field is used as an incident wave in (6.3):

$$
\begin{aligned}
E_r &= \frac{Z_0 Il\cos\theta}{2\pi r^2}\left(1+\frac{1}{jk_0 r}\right)e^{-jk_0 r},\\
E_\theta &= j\frac{Z_0 k_0 Il\sin\theta}{4\pi r}\left(1+\frac{1}{jk_0 r}-\frac{1}{(k_0 r)^2}\right)e^{-jk_0 r},\\
E_\phi &= 0,
\end{aligned}
$$

where

$$
\begin{aligned}
Il &= \text{dipole moment},\\
r &= \text{distance from the dipole source},\\
k_0, Z_0 &= \text{free space wave number and impedance}.
\end{aligned}
$$

The time-averaged radiating power is approximated by using the following far-field

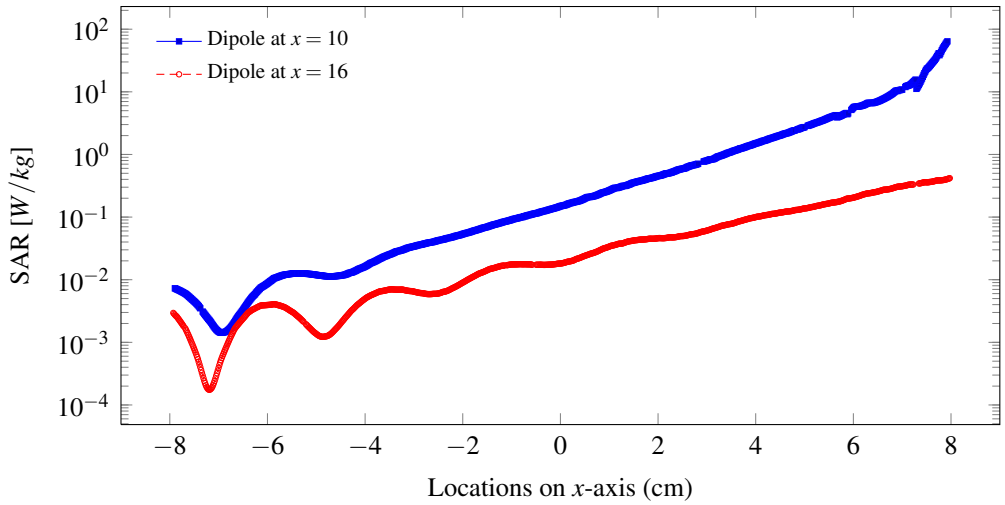


Figure 6.14: Pointwise SAR distribution when the phantom is exposed to dipole sources.

approximation:

$$E_\theta \approx \frac{jk_0 Z_0 Il \sin\theta}{4\pi r} e^{-jk_0 r}, \quad H_\phi \approx \frac{jk_0 Il \sin\theta}{4\pi r} e^{-jk_0 r}.$$

The corresponding radiating power is computed as follows:

$$P = \frac{1}{2}\oint_S Re(\mathbf{E}\times\mathbf{H}^*)\cdot dS = \frac{1}{2}\int_0^{2\pi}\int_0^{\pi} E_\theta H_\phi^* r^2 \sin\theta d\theta d\phi = \frac{Z_0 k_0^2}{12\pi}(Il)^2.$$

Suppose that a cell phone operates in 1 watt of radiating power and we take $Il = 0.016771[A\cdot m]$. With a constant output power, the level of absorbed EM energy in the head is determined by the location of the dipole source. Two dipole sources located at $(10,0,0)$ and $(16,0,0)$ [3] are considered in this experiment. We present the pointwise SAR distribution in Fig. 6.12 and Fig. 6.13. More quantita-

[3]The location is about a quarter wavelength away from the left side of the phantom model.

tively, we plot the SAR over the x-axis in Fig. 6.14. From the graph, we can see that the SAR rapidly decreases as the wave propagates into the head.

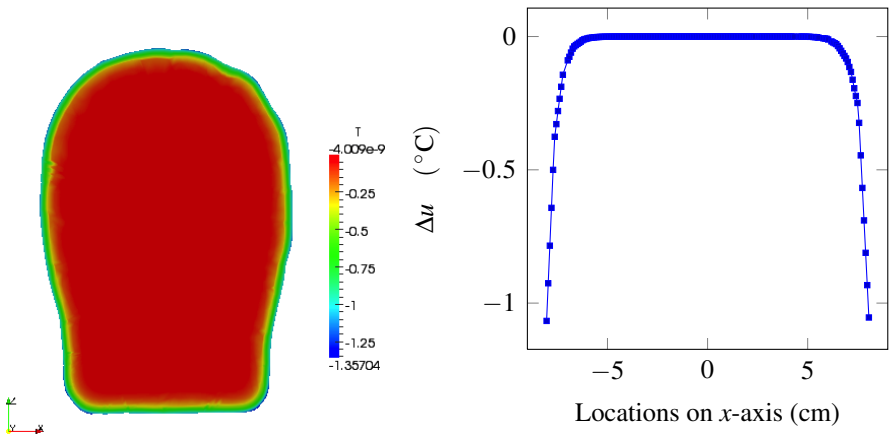
It should be noted that the SAR limit commonly accepted is $2.0[W/kg]$ over average any $10[g]$ tissues [2] and $1.6[W/kg]$ over average any $1[g]$ tissues [3]. Since we evaluate the SAR pointwise, the spatial SAR distribution shown in the figure would be over-estimated compared to what is expected in the averaged values. In this study, we do not consider an averaging scheme as the unstructured tetrahedral mesh is inherently difficult to generate cells by a 10 g or 1g mass. In the transient BHT problem presented next, we only consider a dipole located at $x = 16$.

Time integration. When solving the Pennes BHT equation, we first determine the initial temperature distribution taking into account blood perfusion and metabolism, as well as the convective heat transfer rate in the room temperature. A problem here is that the determination of such physiological quantities is extremely difficult as they are related to complex geometry and temperature inside/outside of the body.

For the simplicity, we assume that the human body maintains a constant temperature and model the metabolic heat generation $\dot{q}_m$ as follows:

$$\int_{\Omega} W_b c_b \left(u_{\text{body}} - u_{\text{a0}}\right) v d\mathbf{x} = \int_{\Omega} \dot{q}_m v d\mathbf{x},$$

where the temperature u_{body} denotes the body temperature. This gives us an almost uniform temperature distribution inside and a thermal boundary layer on the skin. Figure 6.15 shows the steady-state initial temperature distribution in the phantom



Initial temperature distribution Thermal boundary layer

Figure 6.15: Initial temperature distribution in the phantom head model, where the core body temperature is set 37 °C and the room temperature is set 20 °C. The heat convective coefficient is set $h = 10.5[W/(m^2\,^\circ\mathrm{C})]$.

model, where the room temperature is set to be 20 °C and the convective heat transfer coefficient h is taken as $10.5[W/(m^2\,^\circ\mathrm{C})]$ [77]. Note that the mesh is locally refined on the skin surface to resolve the thermal boundary layer.

Starting with the thermal equilibrium status, we advance the time transient BHT equation using the Crank–Nicolson scheme. In FEM, the space and time variables are decoupled and discretized independently to obtain a sequence of algebraic systems. The temperature field θ at the k time iteration is expressed

$$\theta^k(\mathbf{x},t) = \sum_{j=1}^{\mathrm{ndof}} \theta_j^k(t)\phi_j(\mathbf{x}). \tag{6.5}$$

Given the abstract variational form (2.24), the time discretization is described as follows:

$$\left(\frac{\theta^{k+1}-\theta^k}{\Delta t},\psi\right) + b\left(\frac{\theta^{k+1}+\theta^k}{2},\psi\right) = l(\psi),$$

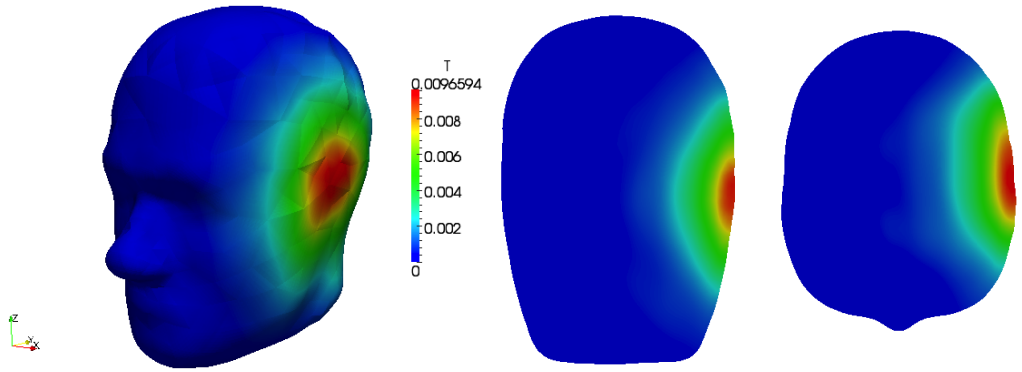


Figure 6.16: Temperature rise after 20 minutes (40 time steps).

where Δt is the time step. It is well known that the scheme is unconditionally stable for diffusion equations [34, 92]. We determine the time step size considering the transient behavior of the temperature in the body. For instance, consider that our head is uniformly heated by SAR $= 1.6[W/kg]$, which is the safety limit suggested in [3]. By assuming that the EM induced heat is entirely used to raise the temperature of the head model, we see that

$$\frac{\Delta\theta}{\Delta t} = \frac{\text{SAR}}{\text{Specific heat of the brain}} \approx 4.3243\text{E-}04.$$

This implies that it will take about 16 minutes to raise the temperature of the brain by 0.25 °C. Here we set the time step to 30 seconds and proceed time-stepping until the temperature field reaches the steady-state.

Figure 6.16 illustrates the solution of the BHT problem taking into account the EM induced heat generation. Volume averaged temperature increase is depicted in Fig. 6.17. The graph shows that the temperature field is quickly elevated in the

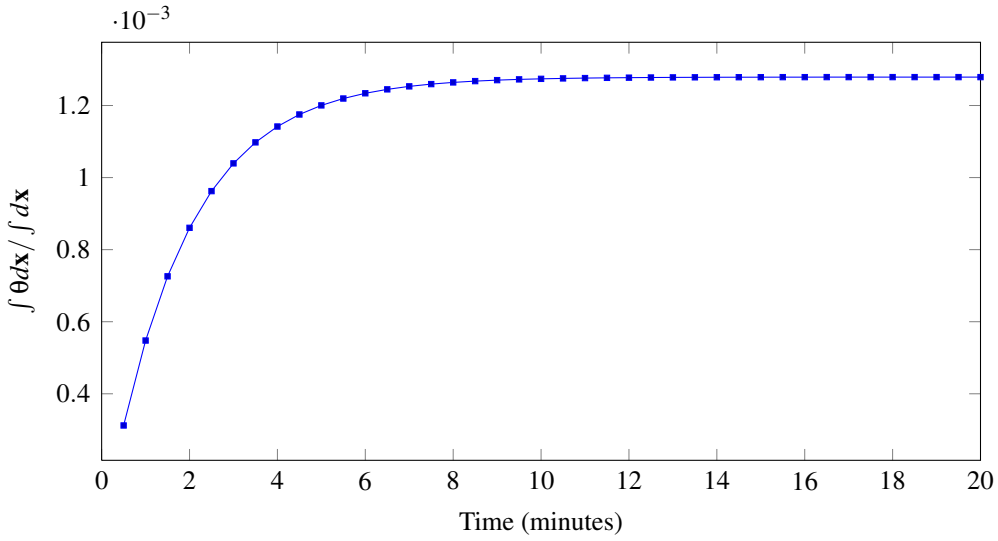


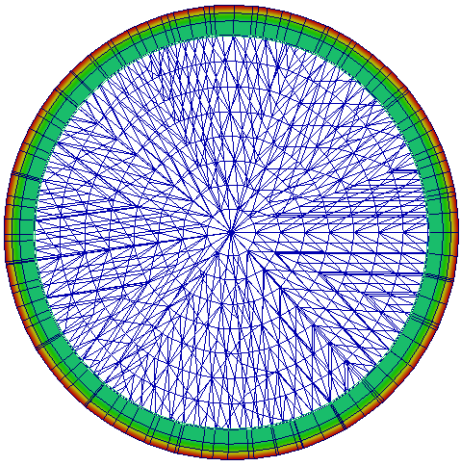
Figure 6.17: Volume averaged temperature rise for 20 minutes.

first six minutes. After then, the temperature field reaches at steady-state temperature distribution in 10 minutes. We compare the result against the solution of the steady-state BHT problem. The non-linear effect due to the temperature dependence of the EM material property is very weak, see (6.4). The EM material properties vary 0.005 for 1 °C degree change [4].

6.4 Four-layered spherical head model

The head model used here consists of a sphere and three layered shells. As depicted in Fig. 6.18, the core sphere represents a brain; it is surrounded by three spherical shells that represent Cerebro-Spinal Fluid (CSF), skull and skin respectively. The sphere comprises tetrahedral elements and three spherical shells are

[4] One degree change is huge considering the SAR level.

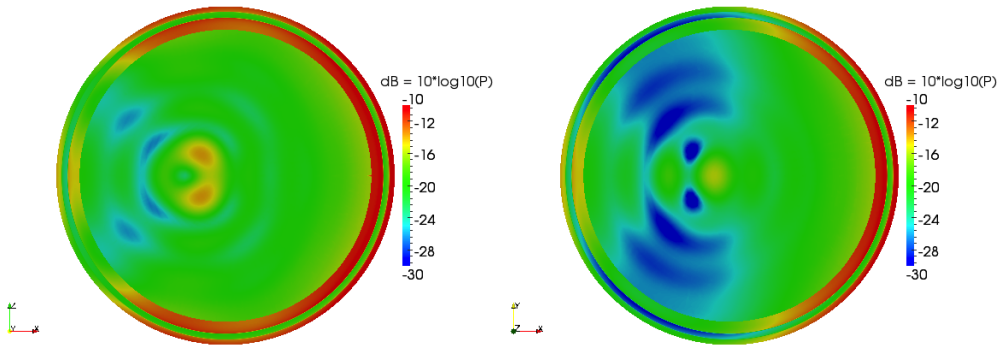


	Thickness(m)	Element type	p	ε_r	ε_s	$\rho[kg/m^3]$
Brain	$r = 0.1$	Tetrahedron	3	4.5805E+01	1.5309E+01	1039
CSF	0.008	Prism	32	6.8638E+01	4.8185E+01	1039
Skull	0.004	Prism	32	1.6621E+01	4.8261E+00	1645
Skin	0.004	Prism	32	4.1405E+01	1.7312E+01	1100

Figure 6.18: Four-layered spherical head model. The core sphere represents a brain; three outer shells represent CSF, skull, and skin respectively.

modeled with prismatic elements. The same incident waves (*i.e.*, a plane wave and a Hertzian dipole) discussed in the previous section are used in this case study. A particular interest lies in the effect of the thin layers on the propagating EM waves. We will compare the numerical result against one obtained from the homogeneous head model.

The plane wave scattering in the four-layered spherical head model is illustrated in Fig. 6.19. One can observe that higher power density appears in the skin and the CSF regions. This is probably due to reflected waves from the skull and the



Slice to y-axis Slice to z-axis)

Figure 6.19: Absorbed power density of the propagating plane waves in the four-layered spherical head model. This figure is scaled by $dB = 10\log_{10}\left(\sigma|E|^2/2\right)$.

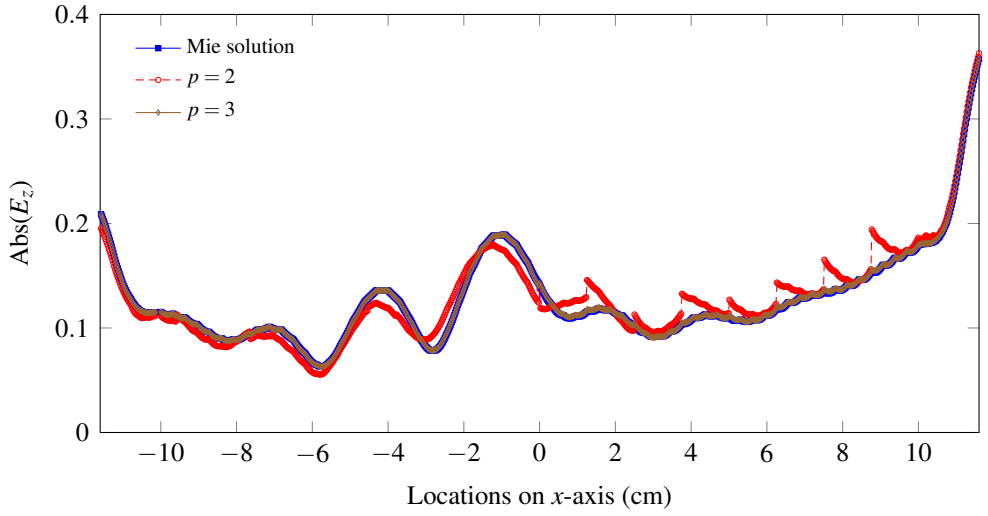


Figure 6.20: Magnitude of the field E_z component propagating in the four-layered spherical head model.

higher conductivity of the CSF. To verify this numerical result, we also compare our four-layered model against the Mie series solution, see Fig. 6.20 and Fig. 6.21. The figures show that our numerical solution approximated with $p = 3$ coincides

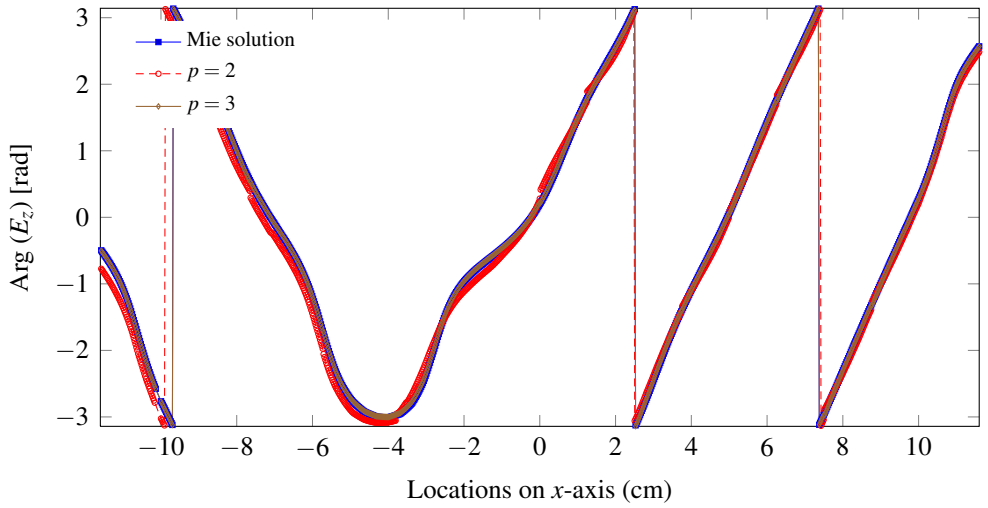
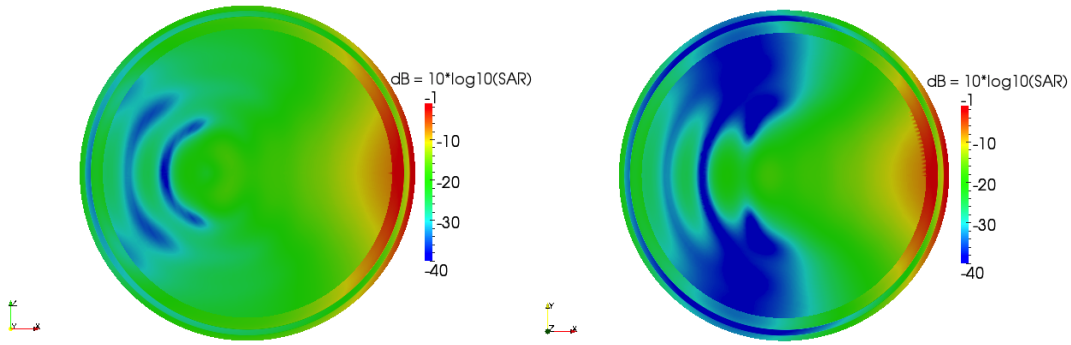


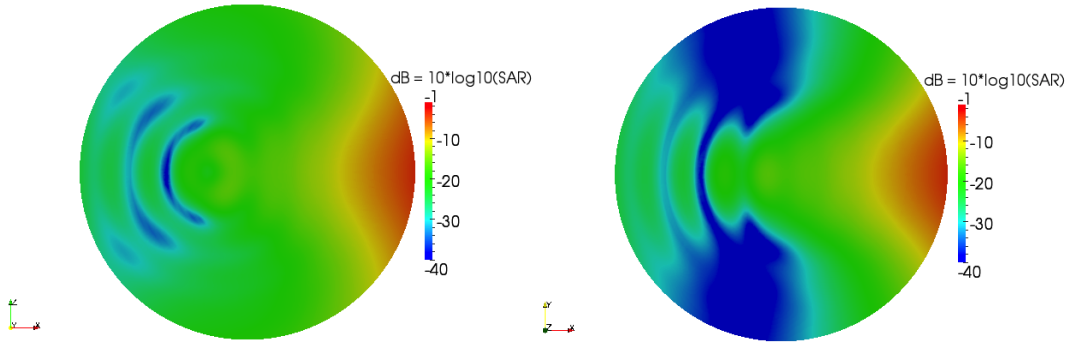
Figure 6.21: Phase angle of the field E_z component propagating in the four-layered spherical head model.

with the analytic Mie series solution. On the other hand, we see the discontinuous electric field when the solution is approximated by $p = 2$. Note that the tangential component of the electric field *i.e.*, E_z should be should be continuous across the material interfaces.

We now consider a case with a dipole source. The dipole is placed at $(0.2m, 0, 0)$, which is about $8.4cm$ away from the skin surface. Figure 6.22 illustrates the different SAR distributions for the layered head model and the homogeneous head model. Higher SAR appears in the CSF region while relatively low level of the SAR is observed in the skull tissue. The maximum SAR for the layered model is computed to be $0.786827[W/kg]$ at the surface of the skin tissue. In addition, almost the same level of SAR is found at the interface between the CSF and skull tissues. This is probably due to the higher contrast in the material properties



Four-layered spherical head model (slice to y and z axes); max SAR = 0.786828 W/kg



Homogeneous spherical head model (slice to y and z axes); max SAR = 0.433603 W/kg

Figure 6.22: SAR distribution in the spherical head models exposed to an infinitesimal dipole radiation (P = 1[W]) located at $x = 20cm$. This figure is scaled by $dB = 10\log_{10}(\mathrm{SAR})$.

between the skull and CSF tissues. In Fig. 6.23, we describe the SAR distribution sampled over the x-axis. The graph captures the characteristic SAR distribution in the thin tissue layers.

As seen in the previous section, the temperature dependence of EM material properties is very weak considering the output power of typical mobile devices. Thus, its non-linear effect is negligible for this case study. We solve a steady-state BHT problem to illustrate the temperature increase induced by the dipole radiation.

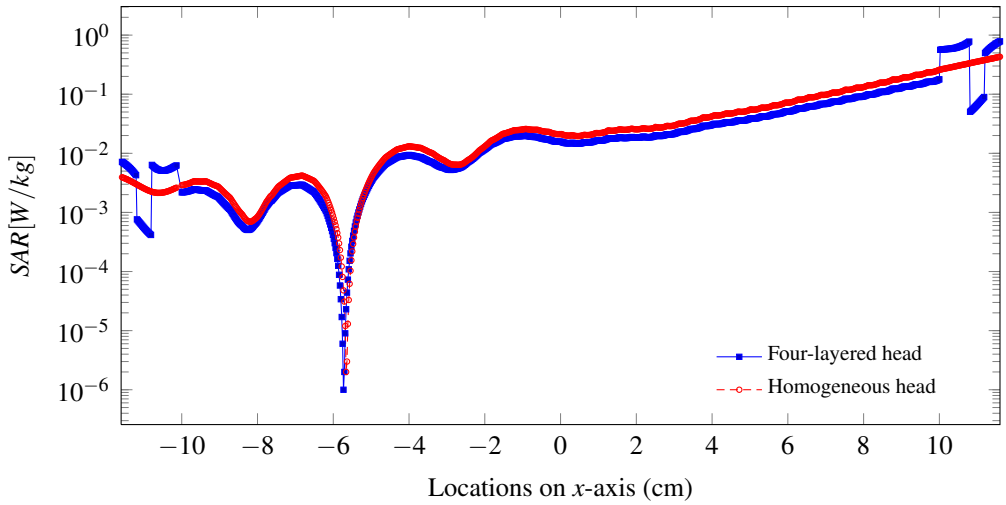
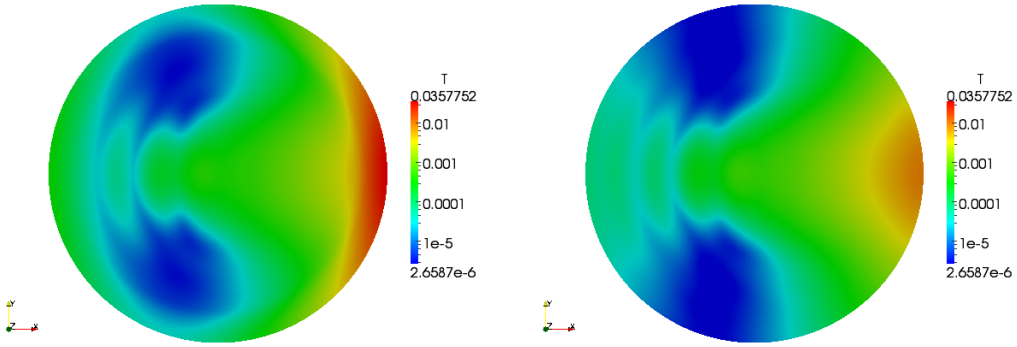


Figure 6.23: Pointwise SAR distribution in the four-layered spherical head model.

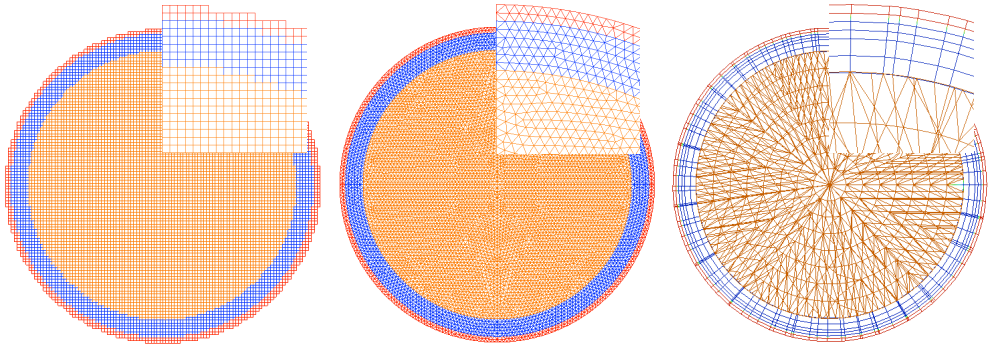


Max Δu = 0.03577 °C

Max Δu = 0.00929 °C

Figure 6.24: Steady-state temperature increase.

We uses thermal properties tabulated in Table 2.2. For this simulation, we used the metabolism data specified in the table. We present the steady-state temperature rise in Fig. 6.24.



FDTD (Hexa=2mm) AIM (Tet=2mm) `hp3d` (Tet=12.5mm $\approx 0.25\lambda$)

	Thickness(m)	ε_r	ε_s
Brain	$r = 0.092$	4.5805E+01	1.5309E+01
Bone	0.008	8.9770E+00	1.8325E+00
Fat	0.004	5.4620E+00	1.0193E+00
Skin	0.004	4.1405E+01	1.7312E+01

EM material properties

Figure 6.25: Different mesh densities adopted by different numerical methods.

6.5 Comparison to other methods: FDTD and AIM

This work has been done within an National Science Foundation (NSF) petascale project (PI: Ali Yilmaz) in collaboration with a team developing an alternative approach *i.e.*, Adaptive Integral Method (AIM). In this section, we briefly compare our solution against ones obtained by other methods [5]: Finite Difference Time Domain (FDTD) [101] and AIM [15, 102].

As discussed in Section 1.2.2, each algorithm has its own advantages and disadvantages. Here we summarize the characteristic features of these methods:

[5] Numerical results used in this comparison are obtained from [37].

- FDTD is easy to implement as the scheme is explicit. Further, the method can be easily extended to a parallel computing environment; the computational domain is easily subdivided and indexed in Cartesian coordinates. On the other hand, the use of regular structured grids limits the flexibility in describing complex geometries and curved boundaries, and this results in a less accurate solution than ones obtained by other methods.

- AIM adopts auxiliary uniform grids that enclose scattering objects to accelerate the solution of Method of Moments (MOM); the method reduces the computational cost of iterative solution methods from $O(N^2)$ to $O(N \log N)$ by using compressed representation of the far-field area using Fast Fourier Transform (FFT).

As seen in Fig. 6.25, different mesh structures are used to describe the four-layered spherical head model. Clearly, the structured voxel mesh used for the FDTD method shows limited capabilities in depicting the curved interfaces and boundaries. The tetrahedral mesh adopted by the AIM method significantly increases the number of DOFs to describe precisely the curved boundary and thin spherical shells. On the other hand, we construct a hybrid mesh consisting of tetrahedral elements with $p = 3$ and prismatic elements with $p = 32$ for the thin layers. The element size in our model corresponds to a quarter wave length for $f = 900MHz$, which is six times larger than the size adopted in other methods.

When we first look at the solutions depicted in Fig. 6.26, they look similar to each other. However, the FDTD solution is less accurate than other solutions

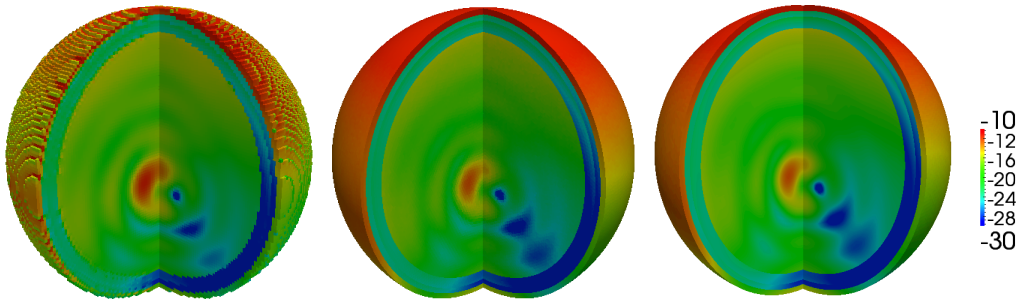


Figure 6.26: Absorbed power density of the propagating plane waves for three different head models. This figure is scaled by $dB = 10\log_{10}\left(\sigma|E|^2/2\right)$.

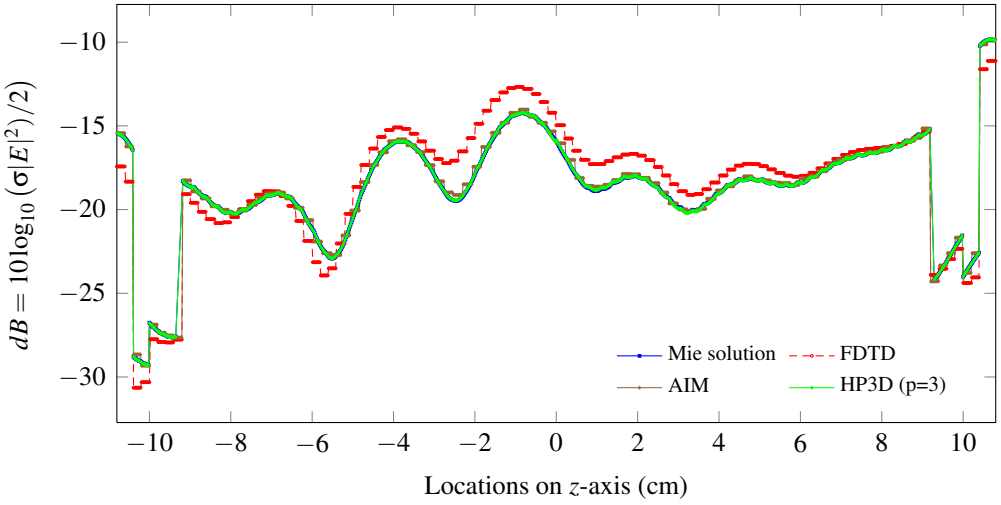


Figure 6.27: Power absorption in the four-layered model, where the solutions are scaled by $dB = 10\log_{10}\left(\sigma|E|^2)/2\right)$.

as demonstrated in Fig. 6.27. This inaccuracy of the method is mainly due to the inaccurate geometry description of the voxel mesh.

6.6 Summary

In this chapter, we applied the developed FE technology to solve Maxwell equations coupled with Pennes BHT equation. We verified our EM solver starting with reproduction of the plane wave solutions including a geometry convergence test. Note that, due to the presence of the element map Jacobian in the Piola transform, $\mathbf{H}(\mathrm{curl})$-conforming discretization is much more sensitive to the regularity of geometry parameterizations than standard H^1-conforming elements.

In the second set of experiments, we solved scattering problems: a plane wave on a PEC and a dielectric sphere. The classical Mie series solutions were used for the code verification. This verification includes our construction of the PML. The numerical examples qualitatively and quantitatively demonstrated that a high quality solution can be delivered by using higher order elements. In addition, we also observed that a precise PML tuning is necessary.

We presented numerical results of the SAR distribution and temperature increase in a head model exposed to EM waves. A G^1-continuous homogeneous head model is constructed in this study using a cloud of fine-grid points. To model cell phone antenna, a Hertzian dipole radiation of output power 1 watt at 900 MHz is placed about 8cm away from the skin. As expected, the higher SAR is found closer to the dipole source; the maximum SAR is detected 0.553604 at skin. The obtained SAR distribution is below the safety limit 1.6 W/kg. The computed corresponding maximum temperature increase in the head is $0.0120\,^{\circ}$C.

We also presented a case study of a four-layered sphere model consisting

of a core brain and surrounding tissues *i.e.*, skin, skull, and CSF. In particular, we demonstrated the effect of those thin-layers on determination of the SAR distribution. The maximum SAR is located at the surface of the skin and the interface between skull and CSF. This is probably due to the reflection/transmission effects caused by the higher contrast of the dielectric material properties in the two media.

Chapter 7

Conclusion

In this chapter, we summarize the results in the present work and suggest opportunities for future research.

7.1 Summary

This dissertation presented the development of a general hp-adaptive Finite Element (FE) technology and a parallel sparse direct solver for solving 3D multi-physics coupled problems. We applied the developed technology to modeling of the Electro-Magnetic (EM) wave propagation in the human head accounting for the resulting dielectric heating effects.

We now present the major achievements and contributions of this dissertation.

FE modeling of EM waves in the human body. The problem is formulated as a coupled form of the time-harmonic Maxwell equations and the transient Pennes BioHeat Transfer (BHT) equation. The EM problem is discretized in space with high order $\mathbf{H}(\text{curl})$-conforming elements while the BHT problem is discretized with H^1-conforming elements. As accuracy of the time discretization (transient effects)

is of little concern, the BHT equation is discretized in time with a simple implicit scheme, the Crank–Nicolson method. Two equations are weakly coupled through the data transfer of Specific Absorption Rate (SAR) induced by EM waves and dependence of EM material properties upon the temperature.

Solving the coupled problem, we implemented $\mathbf{H}(\text{curl})$-conforming elements for tetrahedral and prismatic elements to simulate EM scattering problems. Starting with a simple sphere model, we verified the correctness of the numerical model and developed EM solver against the analytic Mie series solution [95].

Finally, we demonstrated the solution of the coupled problem. Several numerical experiments were conducted; two different head models (*i.e.*, the phantom model and the four-layered sphere model) were mainly used for the study of the temperature rise due to the EM wave radiations. The experimental result indicates that the dependence of the EM material properties on the temperature change is weak and almost negligible in the range of cell phone powers.

***hp*−FE technology.** The `hp3d`, a general purpose three-dimensional *hp*-adaptive FE code, was developed in this work. Departing from the earlier version of `hp3D`, the code is designed to support the exact sequence elements for complex multi-physics problems. The code has a superior ability in describing a complex geometry using the elements of all traditional shapes *i.e.*, hexahedron, prism, pyramid, and tetrahedron.

In particular, the development of the deadlock free refinement algorithms that supports a variety of isotropic and anisotropic refinements for the elements of

all shapes is regarded as a contribution to *hp*-FEM theory and implementations. To the best of our knowledge, no *hp*-code has been developed supporting hybrid mesh refinements for the element of all shapes allowing the mesh one-irregularity. The refinement strategy completes in two steps: 1) local mesh modifications that refine selected elements, and 2) a global closure that recovers the one-irregularity of the mesh. We do not have a formal proof that the algorithm is deadlock-free. However, we verified that the algorithm is deadlock-free through extensive numerical tests.

We also have redesigned and re-implemented the constrained approximation [25, 75] to support constraints on triangular faces. In dealing with triangular faces, a major difficulty arises from the inconsistent system of coordinates between local and global shape functions. This complexity was considerably reduced by adopting the novel idea of orientation embedded shape functions [36, 80]. The local shape functions are always constructed according to the global orientation; a single constraint matrix is only needed for a family of constraints with different orientations. This approach significantly reduces the complexity in implementation.

The developed refinement package was successfully integrated in the `hp3d` code and used as a key component in P. Gatto's project, which researched modeling bone conduction of sound in the human head [35].

The Unassembled HyperMatrix (UHM) solver. As part of the research, we developed a parallel sparse direct solver. In an adaptive context, the developed solver gains higher performance compared to other state-of-the-art direct solvers by reusing the element matrices and their partial factors previously computed [14].

This updating sparse matrix factorization is the first attempt to use the nature of the *hp*-adaptivity in its application context.

We parallelized the UHM solver with two-level task scheduling [47] that matches the two-level parallelism in the multifrontal factorization. We demonstrated that our approach leads to efficient fine-grained task scheduling without the need for explicitly constructing a global Directed Acyclic Graph (DAG).

We proposed a recursive task subdivision scheme to extend our solver to heterogeneous multi-core architectures. By applying our dynamic task subdivision scheme to the UHM solver, we demonstrated that our approach is an effective solution to deliver portable performance on heterogeneous architectures [48, 49].

7.2 Future work

The following list of possible projects is recommended for further research:

- The head problem should be further studied with a more realistic geometry and inhomogeneous material parameters. As realized in the simulation of the layered head model, material heterogeneity can incur complex reflection and transmission of the propagating EM waves. In addition, the complex shape of biological tissues may leads to standing waves that may result in a deep burn inside of tissues.

- This effect can be resolved with finer scale discrete models, which also require the use a large distributed cluster machine. With the increasing computing power at a single node, employing the MPI/OpenMP hybrid model to

parallelize the `hp3d` code should be considered to solve large scale 3D problems.

Bibliography

[1] *Federal Communications Commission - Radio Frequency Safety*. http://transition.fcc.gov/oet/rfsafety/dielectric.html.

[2] Guidelines For Limiting Exposure To Time-Varying Electric, Magnetic, and Electromagnetic Fields. Technical report, International Commission on Non-Ionizing Radiation Protection, 1998.

[3] Radio Frequency (RF) Exposure Compliance of Radiocommunication Apparatus (All Frequency Bands). Technical report, Industry Canada, 2010.

[4] Mark Ainsworth. Discrete dispersion relation for hp-version finite element approximation at high wave number. *SIAM Journal on Numerical Analysis*, 42(2):553–575, 2005.

[5] P. R. Amestoy, I. S. Duff, J. Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2002.

[6] Patrick R. Amestoy, Abdou Guermouche, Jean-Yves L'Excellent, and Stéphane Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel Comput.*, 32(2):136–156, 2006.

[7] Franz Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, September 1991.

[8] I. Babuška and A. K. Aziz. On the angle condition in the Finite Element Method. *SIAM Journal on Numerical Analysis*, 13(2):214–226, 1976.

[9] I. Babuška, M. Griebel, and J. Pitkäranta. The problem of selecting the shape functions for a p-type finite element. *International Journal for Numerical Methods in Engineering*, 28(8):1891–1908, August 1989.

[10] Timothy J. Baker. Mesh generation: Art or science? *Progress in Aerospace Sciences*, 41(1):29–63, January 2005.

[11] Larisa Beilina and Marcus J. Grote. Adaptive hybrid finite element/difference method for maxwells equations. Technical Report September, University of Basel, Switzerland, 2004.

[12] Jean-Pierre Berenger. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of computational physics*, 114(2):185–200, 1994.

[13] Marsha J. Berger and Randall J. LeVeque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM Journal on Numerical Analysis*, 35(6):2298–2316, 1998.

[14] Paolo Bientinesi, Victor Eijkhout, Kyungjoo Kim, Jason Kurtz, and Robert van de Geijn. Sparse Direct Factorizations through Unassembled Hyper-

Matrices. *Computer Methods in Applied Mechanics and Engineering*, 199:430–438, 2010.

[15] E. Bleszynski, M. Bleszynski, and T. Jaroszewicz. AIM: Adaptive integral method for solving large-scale electromagnetic scattering and radiation problems. *Radio Science*, 31(5):1225–1251, 1996.

[16] Alfredo Buttari, Julien Langou, Jakub Kurzak, and Jack Dongarra. A Class of Parallel Tiled Linear Algebra Algorithms for Multicore Architectures. *Parallel Computing*, 35(1):38–53, 2009.

[17] P. Carnevali, R. B. Morris, Y. Tsuji, and G. Taylor. New basis functions and computational procedures for p-version finite element analysis. *International journal for numerical methods in engineering*, 36(22):3759–3779, 1993.

[18] W. Cecot, W. Rachowicz, and L. Demkowicz. An hp-Adaptive Finite Element Method for Electromagnetics. Part 3: A Three-dimensional Infinite Element for Maxwell's Equations. *International Journal for Numerical Methods in Engineering*, 57(7):899–921, June 2003.

[19] Ernie Chan, Field G. van Zee, Enrique S. Quintana-Orti, Gregorio Quintana-Orti, and Robert A. van de Geijn. Satisfying your dependencies with Supermatrix. In *2007 IEEE International Conference on Cluster Computing*, pages 91–99. IEEE, 2007.

[20] Alice J. Chen and Yannis Kallinderis. Adaptive hybrid (prismatic-tetrahedral) grids for incompressible flows. *International Journal for Numerical Methods in Fluids*, 26(9):1085–1105, May 1998.

[21] Michael M. Chen and Kenneth R. Holmes. Microvascular constributions in tissue heat transfer. *Annals of the New York Academy of Sciences*, 335:137–150, 1980.

[22] Weng Cho Chew and William H. Weedon. A 3D perfectly matched medium from modified Maxwell's equations with stretched coordinates. *Microwave and optical technology*, 7(13):599–604, 1994.

[23] L. Demkowicz, A. Bajer, W. Rachowicz, and K. Gerdes. 3D hp-Adaptive Finite Element Package Fortran 90 Implementation (3Dhp90). Technical report, TICAM Report 99-29, The University of Texas at Austin, 1999.

[24] L. Demkowicz, P. Gatto, W. Qiu, and A. Joplin. G1-Interpolation and geometry reconstruction for higher order finite elements. *Computer Methods in Applied Mechanics and Engineering*, 198(13-14):1198–1212, March 2009.

[25] L. Demkowicz, J. T. Oden, W. Rachowicz, and O. Hardy. Toward A Universal hp Adaptive Finite Element Strategy, Part 1. Constrained Approximation and Data Structure. *Comput. Methods. Appl. Mech. and Engg*, 77:79–112, 1989.

[26] Leszek Demkowicz. *Computing with hp-Adaptive Finite Elements, vol 1, One and Two Dimensional Elliptic and Maxwell Problems*. Chapman &

HallCRC, 2007.

[27] Leszek Demkowicz, Jason Kurtz, David Pardo, Maciej Paszynski, Waldemar Rachowicz, and Adam Zdunek. *Computing with Hp-Adaptive Finite Elements, Vol. 2: Frontiers Three Dimensional Elliptic and Maxwell Problems with Applications*. Chapman & HallCRC, 2007.

[28] Leszek Demkowicz, David Pardo, and Waldek Rachowicz. 3D hp-Adaptive Finite Element Package (3Dhp90) Version 2.0: The Ultimate Data Structure for Three Dimensional, Anisotropic hp Refinements. Technical Report TICAM Report 02-24, TICAM Report 02-24, The University of Texas at Austin, 1999.

[29] Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and Iain Duff. A set of level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Soft.*, 16(1):1–17, March 1990.

[30] I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear. *ACM Transactions on Mathematical Software (TOMS)*, 9(3):302–325, 1983.

[31] Alejandro Duran, J. Corbalán, and E. Ayguadé. Evaluation of OpenMP task scheduling strategies. In *Proceedings of the 4th international conference on OpenMP in a new era of parallelism*, pages 100–110. Springer-Verlag, 2008.

[32] Bjorn Engquist and Andrew Majdat. Absorbing boundary conditions for

numerical simulation of waves. *Applied Mathematics Sciences*, 74(5):1765–1766, 1977.

[33] A. M. Erisman and J. K. Reid. Monitoring the stability of the triangular factorization of a sparse matrix. *Numerische Mathematik*, 22(3):183–186, June 1974.

[34] David Fuentes, Yusheng Feng, Andrew Elliott, Anil Shetty, Roger J. McNichols, J. Tinsley Oden, and R. Jason Stafford. Adaptive real-time bioheat transfer models for computer-driven MR-guided laser induced thermal therapy. *IEEE transactions on bio-medical engineering*, 57(5):1024–30, May 2010.

[35] Paolo Gatto. *Modeling bone conduction of sound in the human head using hp-finite elements*. PhD thesis, The University of Texas at Austin, 2012.

[36] Paolo Gatto and Leszek Demkowicz. Construction of H1-conforming hierarchical shape functions for elements of all shapes and transfinite interpolation. *Finite Elem. Anal. Des.*, 46(6):474–486, June 2010.

[37] Cemil S. Geyik, Fangzhou Wei, Jackson W. Massey, and Ali E. Yilmaz. FDTD vs. AIM for bioelectromagnetic analysis. In *Proceedings of the 2012 IEEE International Symposium on Antennas and Propagation*, pages 1–2. Ieee, July 2012.

[38] Abdou Guermouche, Jean-Yves L'Excellent, and Gil Utard. Impact of reordering on the memory of a multifrontal solver. *Parallel Computing*,

29(9):1191–1218, September 2003.

[39] Han Guo, Jun Hu, Hanru Shao, and Zaiping Nie. Hierarchical Matrices Method and Its Application in Electromagnetic Integral Equations. *International Journal of Antennas and Propagation*, 2012:1–9, 2012.

[40] Fred G. Gustavson, Isak Jonsson, Bo Kå gström, and Per Ling. Towards peak performance on hierarchical SMP memory architectures - new recursive blocked data formats and BLAS. In *Parallel Processing for Scientific Computing*, pages 1–4, 1999.

[41] Arthur W. Guy. Electromagnetic Fields and Relative Heating Patterns Due to a Rectangular Aperture Source in Direct Contact with Bilayered Biological Tissue. *IEEE Transactions on Microwave Theory and Techniques*, MTT-19(2):214–223, 1971.

[42] Roger F. Harrington. *Field computation by moment methods*. Wiley-IEEE Press, 1968.

[43] Tamer S. Ibrahim, Doney Abraham, and Robert L. Rennaker. Electromagnetic power absorption and temperature changes due to brain machine interface operation. *Annals of Biomedical Engineering*, 35(5):825–834, 2007.

[44] Jian-Ming Jin. *Theory and computation of electromagnetic fields*. Wiley, 2011.

[45] Curtis C. Johnson and Arthur W. Guy. Nonionizing electromagnetic wave effects in biological materials and systems. *Proceedings of the IEEE*, 60(6):692–718, 1972.

[46] George Karypis and Vipin Kumar. METIS : A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. Technical report, University of Minnesota, 1998.

[47] Kyungjoo Kim and Victor Eijkhout. A Parallel Sparse Direct Solver via Hierarchical DAG Scheduling. Technical report, TR-12-05, Texas Advanced Computing Center, The University of Texas at Austin, 2012.

[48] Kyungjoo Kim and Victor Eijkhout. Scheduling a Parallel Sparse Direct Solver to Multiple GPUs. In *The 14th IEEE Workshop on Parallel and Distributed Scientific and Engineering Computing*, 2013.

[49] Kyungjoo Kim, Victor Eijkhout, and Robert A. van de Geijn. Dense Matrix Computation on a Heterogenous Architecture: A Block Synchronous Approach. Technical report, TR-12-04, Texas Advanced Computing Center, The University of Texas at Austin, 2012.

[50] Benjamin S. Kirk, John W. Peterson, Roy H. Stogner, and Graham F. Carey. libMesh : a C++ library for parallel adaptive mesh refinement/coarsening simulations. *Engineering with Computers*, 22(3-4):237–254, November 2006.

[51] Tim Kröger and Tobias Preusser. Stability of the 8-tetrahedra shortest-interior-edge partitioning method. *Numer. Math.*, 109(23):435–457, April 2008.

[52] Michal Kížek and Theofanis Strouboulis. How to generate local refinements of unstructured tetrahedral meshes satisfying a regularity ball condition. *Numerical Methods for Partial Differential Equations*, 13(2):201–214, March 1997.

[53] Mariya Lazebnik, Mark C. Converse, John H. Booske, and Susan C. Hagness. Ultrawideband temperature-dependent dielectric properties of animal liver tissue in the microwave frequency range. *Physics in Medicine and Biology*, 51(7):1941–55, April 2006.

[54] A. Liu and B. Joe. Quality local refinement of tetrahedral meshes based on 8-subtetrahedron subdivision. *Mathematics of Computation*, 65(215):1183–1200, July 1996.

[55] Joseph W. H. Liu. On threshold pivoting in the multifrontal method for sparse indefinite systems. *ACM Transactions on Mathematical Software (TOMS)*, 13(3):250–261, 1987.

[56] Sai Huen Lo. Volume discretization into tetrahedra-II. 3D triangulation by advancing front approach. *Computers and Structures*, 39(5):501–511, 1991.

[57] Tze Meng Low and Robert A. van de Geijn. An API for manipulating matrices stored by blocks. Technical report, FLAME Working Note 12,

TR-2004-15, The University of Texas at Austin, 2004.

[58] Xiao-juan Luo, Mark S. Shephard, and Jean-Francois Remacle. P-version mesh generation issues. In *11th International Meshing Roundtable, Sandia National Laboratories*, pages 343–354, 2002.

[59] Xiaojuan Luo, Mark S. Shephard, and Jean-Francois Remacle. The influence of geometric approximation on the accuracy of high order methods. Technical report, Rensselaer Polytechnic Institute, 2002.

[60] Pawel J. Matuszyk and Leszek F. Demkowicz. Parametric finite elements, exact sequences and perfectly matched layers. *Computational Mechanics*, 51(1):34–45, 2013.

[61] J. M. Melenk and S. Sauter. Wave number explicit convergence analysis for Galerkin discretizations of the Helmholtz equation. *SIAM Journal on Numerical Analysis*, 49(3):1210–1243, 2011.

[62] Christian Michler, Leszek Demkowicz, Jason Kurtz, and David Pardo. Improving the performance of perfectly matched layers by means of hp-adaptivity. *Numerical Methods for Partial Differential Equations*, 23(4):832–858, 2007.

[63] W. W. Mumford. Some technical aspects of microwave radiation hazards. In *Proceedings of the IRE*, pages 427–447, 1960.

[64] J. C. Nedelec. Mixed Finitc Elements in R3. *Numerische Mathematik*, 1(3):315–341, 1980.

[65] J. C. Nedelec. A New Family of Mixed Finite Elements in R3. *Numerische Mathematik*, 81:57–81, 1986.

[66] Michal Okoniewski, Ewa Okoniewska, and Maria A. Stuchly. Three-dimensional subgridding algorithm for FDTD. *Antennas and Propagation, IEEE Transactions on*, 45(3):422–429, 1997.

[67] Stephen L. Olivier and Jan F. Prins. Comparison of OpenMP 3.0 and Other Task Parallel Frameworks on Unbalanced Task Graphs. *International Journal of Parallel Programming*, 38(5-6):341–360, June 2010.

[68] OpenMP Architecture Review Board. *OpenMP Application Program Interface, Version 3.0*. `http://www.openmp.org`, 2008.

[69] Steven J. Owen. A survey of unstructured mesh generation technology. In *7th International Meshing Roundtable*, pages 239–267, 1998.

[70] Steven J. Owen and David R. White. Mesh-based geometry: A systematic approach to constructing geometry from a finite element mesh. *Proc. 10th Int. Meshing Roundtable, Sandia report*, pages 1–17, 2001.

[71] A. K. Patra, A. Laszloffy, and J. Long. Data Structures and Load Balancing for Parallel Adaptive hp Finite-Element Methods. *Computers and Mathematics with Applications*, 46(1):105–123, 2003.

[72] Harry H. Pennes. Analysis of tissue and arterial blood temperatures in the resting human forearm. *Journal of applied physiology*, 1(2):93–122, 1948.

[73] Angel Plaza and Maria-Cecilia Rivara. Mesh refinement based on the 8-tetrahedra longest-edge partition. In *Proceedings, 12th International Meshing Roundtable*, pages 67–78, 2003.

[74] Gregorio Quintana-Ortí, Enrique S. Quintana-Ortí, Robert A. van de Geijn, Field G. van Zee, and Ernie Chan. Programming matrix algorithms-by-blocks for thread-level Parallelism. *ACM Transactions on Mathematical Software*, 36(3):1–26, July 2009.

[75] Waldek Rachowicz and Leszek Demkowicz. An hp-adaptive finite element method for electromagnetics Part 1: Data structure and constrained approximation. *Computer Methods in Applied Mechanics and Engineering*, 187(1-2):307–335, June 2000.

[76] Werner C. Rheinboldt and Charles K. Mesztenyi. On a Data Structure for Adaptive Finite Element Mesh Refinements. *ACM Transactions on Mathematical Software*, 6(2):166–187, June 1980.

[77] A. I. Sabbah and N. I. Dib. SAR and Temperature Elevation in a Multi-Layered Human Head Model Due to an Obliquely Incident Plane Wave. *Progress In Electromagnetics Research*, 13:95–108, 2010.

[78] Olaf Schenk and Klaus Gärtner. Solving unsymmetric sparse systems of linear equations with PARDISO. In *Proceedings of the International Conference on Computational Science-Part II*, pages 335–363. Springer-Verlag, 2002.

[79] Olaf Schenk and Klaus Gärtner. On fast factorization pivoting methods for sparse symmetric indefinite systems. *Electronic Transactions on Numerical Analysis*, 23:158–179, 2006.

[80] Joachim Schöberl and Sabine Zaglmayr. High order Nédélec elements with local complete sequence properties. *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, 24(2):374–384, 2005.

[81] A. Schröder. Constrained approximation in hp-FEM: Unsymmetric subdivisions and multi-level hanging nodes. *Spectral and High Order Methods for Partial Differential Equations*, 76:317–325, 2011.

[82] Sanjiv Shah, Grant Haab, Paul Petersen, and Joe Throop. Flexible control structures for parallelism in OpenMP. *Concurrency: Practice and Experience*, 12(12):1219–1239, 2000.

[83] M. S. Shephard, S. Dey, and J. E. Flaherty. A straight forward structure to construct shape functions for variable p-order meshes. *Computer Methods in Applied Mechanics and Engineering*, 147(3-4):209–233, 1997.

[84] Mark S. Shephard and Marcel K. Georges. Automatic three-dimensional mesh generation by the finite octree technique. *Int. J. Num. Meth. Eng.*, 32(4):709–749, 1991.

[85] Mark S. Shephard, Mark A. Yerry, and Peggy L. Baehmann. Automatic

mesh generation allowing for efficient a priori and a posteriori mesh refinement. *Comput. Methods Appl. Mech. Eng.*, 55:161–180, 1986.

[86] J. R. Shewchuk. What Is a Good Linear Finite Element? Interpolation, Conditioning, Anisotropy, and Quality Measures. In *11th International Meshing Roundtable, Sandia National Laboratories*, 2002.

[87] Hang Si. TetGen: A quality tetrahedral mesh generator and three-dimensional delaunay triangulator. Technical report, 2006.

[88] P. Silvester. Finite element solution of homogeneous waveguide problems. *Alta Frequenza*, 38:313–317, 1969.

[89] Pavel Solin, Jakub Cerveny, and Ivo Dolezel. Arbitrary-level hanging nodes and automatic adaptivity in the hp-FEM. *Mathematics and Computers in Simulation*, 77(1):117–132, February 2008.

[90] Jiming Song, Cai-Cheng Lu, and Weng Cho Chew. Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects. *Antennas and Propagation, IEEE*, 45(10):1488–1493, 1997.

[91] Christian Terboven, Dieter an Mey, Dirk Schmidl, Henry Jin, and Thomas Reichstein. Data and thread affinity in OpenMP programs. In *Proceedings of the 2008 workshop on Memory access on future processors*, pages 377–384, New York, USA, 2008.

[92] J. W. Thomas. *Numerical partial differential equations: finite difference methods*. Springer, 1995.

[93] Vinod Valsalam and Anthony Skjellum. A framework for high-performance matrix multiplication based on hierarchical abstractions, algorithms and optimized low-level kernels. *Concurrency and Computation: Practice and Experience*, 14(10):805–839, August 2002.

[94] S. Weinbaum and L. M. Jiji. A new simplified bioheat equation for the effect of blood flow on local average tissue temperature. *Journal of biomechanical engineering*, 1985.

[95] W. J. Wiscombe. Improved Mie scattering algorithms. *Applied optics*, 19(9):1505–1509, 1980.

[96] Eugene H. Wissler. Pennes 1948 paper revisited. *Journal of applied physiology*, 85:35–41, 1998.

[97] Dong Xue. *Control of geometry error in hp finite element (FE) simulations of electromagnetic (EM) waves*. PhD thesis, The University of Texas at Austin, 2005.

[98] Dong Xue, Leszek Demkowicz, and Chandrajit Bajaj. Reconstruction of G1 surfaces with biquartic patches for hp-FE simulations. In *13th International Meshing Roundtable*, volume 13, pages 323–332, 2004.

[99] Dong Xue, Leszek Demkowicz, and Adam Zdunek. An Interface Between Geometrical Modeling Package (GMP) and Mesh-Based Geometry (MBG). Technical report, ICES 03-20, Texas Institute for Computational and Engineering Science, The University of Texas at Austin, Austin, 2000.

[100] Soji Yamakawa and Kenji Shimada. Converting a tetrahedral mesh to a prism-tetrahedral hybrid mesh for FEM accuracy and efficiency. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, volume 1, pages 287–294, 2008.

[101] K. Yee. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *Antennas and Propagation, IEEE Transactions on*, 14(3):302–307, January 1966.

[102] Ali E. Yilmaz, Jian-Ming Jin, and Eric Michielssen. Time Domain Adaptive Integral Method for Surface Integral Equations. *IEEE Transactions on Antennas and Propagation*, 52(10):2692–2708, October 2004.

[103] Ali E. Yilmaz, Daniel S. Weile, Jian-Ming Jin, and Eric Michielssen. A hierarchical FFT algorithm (HIL-FFT) for the fast analysis of transient electromagnetic scattering phenomena. *Antennas and Propagation, IEEE Transactions onand Propagation, IEEE*, 50(7):971–982, 2002.

[104] A. R. Zakharian, M. Brio, and J. V. Moloney. FDTD based second-order accurate local mesh refinement method for Maxwell's equations in two space dimensions. *Communications in Mathematical Sciences*, 2(3):497–513, 2004.