Technische Universität München
Lehrstuhl für Computation in Engineering

The Finite Cell Method for Geometry-Based Structural Simulation

Zhengxiong Yang

Vollständiger Abdruck der von der Fakultät für Bauingenieur- und Vermessungswesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr.-Ing. habil. M. Manhart
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr.rer.nat. E. Rank
2.	Univ.-Prof. Dr.-Ing. habil. A. Düster, Technische Universität Hamburg-Harburg
3.	Prof. Dr. J. Parvizian, Isfahan University of Technology, Isfahan, Iran

Die Dissertation wurde am 16.03.2011 bei der Technischen Universität München eingereicht und durch die Fakultät für Bauingenieur- und Vermessungswesen am 22.06.2011 angenommen.

Abstract

The finite cell method is a fictitious domain method combined with p-version high-order polynomial basis functions. This method enables the utilization of easily generated structured grids to replace regular FE meshes. With trivial effort in mesh generation the major computational cost lies in the computation of stiffness matrices. This thesis presents a CT-derived data specific integration scheme which accelerates the stiffness matrices computation by precomputation with respect to material constants and voxel dimensions. With this scheme applied to solve three-dimensional isotropic linear elastic problems, a remarkable reduction in computational time is achieved. Moreover a good accuracy is also obtained and verified by several numerical examples. The high efficiency and accuracy of this integration scheme enable establishment of a prototype of an interactive surgical planning platform which allows for predicting and monitoring in-vivo bone-implant stress distribution in real-time via computational steering.

Preface

The work presented in this dissertation was developed during my PhD study majored in numerical simulation at the Chair for Computation in Engineering at Technische Universität München (October 2006 - February 2011). This work was funded by the Siemens, Cooperate Technology PP2 and the International Graduate School of Science and Engineering, IGSSE under the project "Computational steering for Orthopaedics". This support is gratefully acknowledged.

I would like to thank all who contributed towards the accomplishment of this work. Firstly, I truly thank Prof. Dr. Ernst Rank for being the strict yet great mentor who has taught me, both consciously and unconsciously, how to explain things simply and clearly. His ideas and suggestions had guided me through my research and had profoundly influenced my working direction. Beside his support in academy, I am also grateful for his help and care in the daily life. My gratefully and sincerely thank goes to Prof. Dr. Albert Gilg for the financial support from Siemens AG, CT PP2 by an IGSSE scholarship, the entire work would not have been possible without this support. I am also greatly indebted to Prof. Dr.-Ing. habil. Alexander Düster, my second supervisor, because of his skillful supervision, because of his confidence in me, and because of his rigorous and structured working method that has deeply influenced my way of working. It has been a pleasure to share new ideas and discuss with him. A big thank goes to my third supervisor Dr.-Ing. Stefan Kollmannsberger who had been my supervisor during my master work and has continued to supervise me for part of my PhD work. I deeply appreciate his effort and encouragement in guiding me through one of the most tedious part of the project, as well as his patience and kindness during the correction of this dissertation. I wish to thank Prof. Zohar Yosibash for his generous support with abundant expertise in biomechanical simulation of human hard and soft tissues. I also thank Prof. Jamshid Parvizian, one of the inventors of the finite cell method, for his help on me and his long-term concern over the development of the method. Also I would like to thank my supervisor of the IGSSE project, Dr.-Ing. Martin Ruess who has contributed a lot in the organization towards the achievement of the project goals. Furthermore, I wish to thank Dr. Utz Wever, my supervisor from Siemens, CT PP2, for his constant concern of the project and continuous support and inspiration on my work.

Furthermore, I would like to express my gratitude to all my colleagues at the chair for Computation in Engineering at Technische Universität München for the various ways in which they have supported me: A special thank you goes out to Dr.-Ing Dmitry Ledentsov, my former colleague from whom I learned a lot of useful software and programming skills. His enthusiasm about science had also enlightened my aspiration to knowledge. I also thank Mr. Martin Schlaffer for his invaluable technical support on the computer system. Mr. David Franke and Michael Pfaffinger had been great colleagues next door for helping me out with many things. I am also grateful to my two colleagues of the IGSSE project Mr. Eduardo Grande and Mr. Christian Dick for their effort in accomplishing the project and valuable help.

Finally, I would like to thank my parents for their love, understanding and support on my study over the last twenty years. And last but no least, a special thank to my wife Miao Yu,

without whom I would be a completely different person today. Her endless love and tender care supported me over my six years stay in Germany, my PhD work would have been much more difficult without her love. In addition I also appreciate her for helping me with part of the correction of this dissertation.

Munich, Germany, Febuary 2011
Zhengxiong Yang

Contents

Chapter 1

Introduction

As the finite element analysis methods is frequently implemented in the biomechanical analysis, the disadvantages of standard FEM in modeling and meshing the biological structures start to emerge. It is because of the geometrical differences between individuals, the discontinuity of material properties, and also the complexity of structures. For instance, in the mechanical analysis of a human femur with the finite element method [1], the construction of a volume mesh is labor intensive.

One possibility of bypassing the computationally expensive mesh generation step is to use the finite cell method (FCM) [2, 3] which is a fictitious domain method combined with high-order shape functions. The basic idea is to extend the partial differential equation beyond the physical domain up to the boundary of a fictitious domain which is rectangular and can be easily meshed with structured grids, so called cells. Within each cell, high-order shape functions are applied to approximate the displacement field. Strains and stresses are also computed based on high-order shape functions. However, the primary field variables are the displacements. This method shows exponential rate of convergence for smooth problems and even good accuracy for problems with singularities [2, 3]. The main computational effort of this method lies in performing numerical integration of stiffness matrices, e.g. a dense set of integration points is needed when Gaussian quadrature is applied. Thus, the necessity of speeding up the stiffness matrices computation emerges in simulations where computational speed is demanding. A CT-derived data specific fast integration scheme has been developed and is presented in this dissertation. The basic idea of this scheme is to precompute the stiffness matrices with respect to the two Lamé constants and CT voxel dimensions so as to avoid performing Gaussian integration during run-time. This scheme shows both good accuracy and enormously high efficiency in voxel-based numerical analyses demonstrated by several numerical examples. The FCM with fast integration can be applied to solve various problems which when solved with the standard FEM are time-consuming or even not feasible. One of the applications is a computational steering system for patient-specific pre-operative surgical planning.

Fast and reliable methods for predicting and monitoring in-vivo bone strength are of major importance in clinical applications such as fracture fixation or endoprostheses for joint replacement. Furthermore, the development of implants calls for highly efficient and robust analysis tools which allow, during a design loop, matching the mechanical properties of implants with those of the individual bone in order to avoid adaptive remodeling with cortical thinning and

increased porosity of the bone. The main objective of this dissertation is to develop and implement a prototype of such an analysis tool. The simulation kernel of this tool is powered by the FCM with fast integration, which is extended to fulfill the requirement of the steering system, while the visualization kernel is developed by the collaborative partner and is coupled with the simulation kernel via high speed internet connection. The established surgical planning system shows a good performance and opens new means to the surgeons who can perform pre-operative planning with help of the real-time stress visualization.

The outline of this dissertation is as follows:

Chapter 2 introduces the biomechanical analysis of human bone as to provide preparation and groundwork for *Chapter 4* and *5*.

Chapter 3 contains an introduction to the p-version finite element method. Starting from the basic principles in three-dimensional linear elasticity the weak formulation is derived. This is followed by the presentation of one and three-dimensional hierarchical shape functions.

Being a new method that combines the idea of the fictitious domain method with the p-version finite element method, the finite cell method (FCM) is introduced in *Chapter 4*. After the introduction of basic formulations, the standard approach for numerical integration in the FCM is elaborated. As a special extension to the standard FCM, a CT-derived data specific fast integration scheme for three-dimensional problems of linear elastostatics is then presented. The accuracy and efficiency of this new integration scheme is verified by four numerical examples of three-dimensional linear elasticity.

In *Chapter 5* starting from the introduction of hip replacement and the concept of computational steering, the methodology of establishing the surgical planning system is addressed in detail. A demonstration example of using the surgical planning system is given to show one possible application of the system in helping surgeons' decision making. Furthermore, the accuracy of the surgical planning system is validated by an experiment given at the end of this chapter.

Finally, conclusions are drawn in *Chapter 6*.

Chapter 2

Biomechanical analysis of human femur

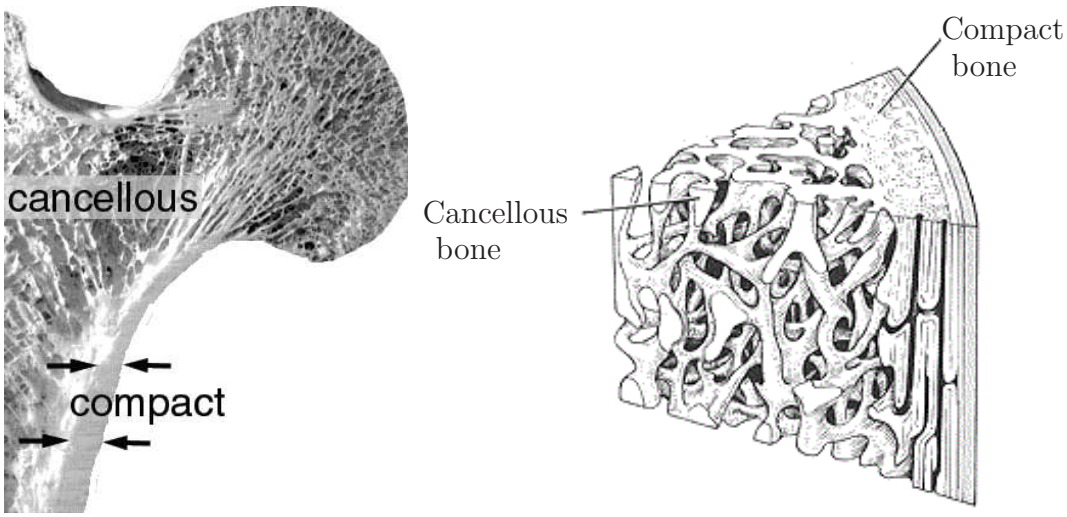
The mechanical response of the human femur has been an on-going research topic during the past centuries. The development of modern theories of mechanics and computer-aided computational tools, like finite element analysis (FEA), enables an in-depth study on the femur properties and provides a convenient yet reliable approach to predict the femur's mechanical response. In this dissertation the finite cell method is employed in mechanical analysis of the femur as well as the pre-selection of implants in surgical planning. As an introduction to the biomechanical aspect of the topic, this chapter closely follows the description in [4, 5] and aims at providing some basic knowledge of the mechanical properties of the human bone tissue (especially the femur) and briefly introducing several commonly used finite element analysis methods in femur analysis.

2.1 Bone physiology and anatomy

The skeletal system is a major constituent of the human body and consists of bones and connective tissues, e.g., ligaments, tendons, muscles and cartilage. As porous mineralized structures, bones are rigid and living organs that serve for moving, supporting and protecting interior organs of the body. Bones are made up of marrow, blood vessels, epithelium, nerves and the major composition: bone tissues, which are the mineral matrix that compose the rigid parts of the organ.

Bone tissue consists of cells embedded in a fibrous organic matrix which is primarily collagen (90%) and 10% amorphous ground substance. Based on its structural, mechanical and metabolic function, bone tissue can be classified into two types, cortical bone and trabecular bone, also referred to as compact bone and cancellous bone [6]. Figure 2.1 depicts the cancellous and compact structure of a bone.

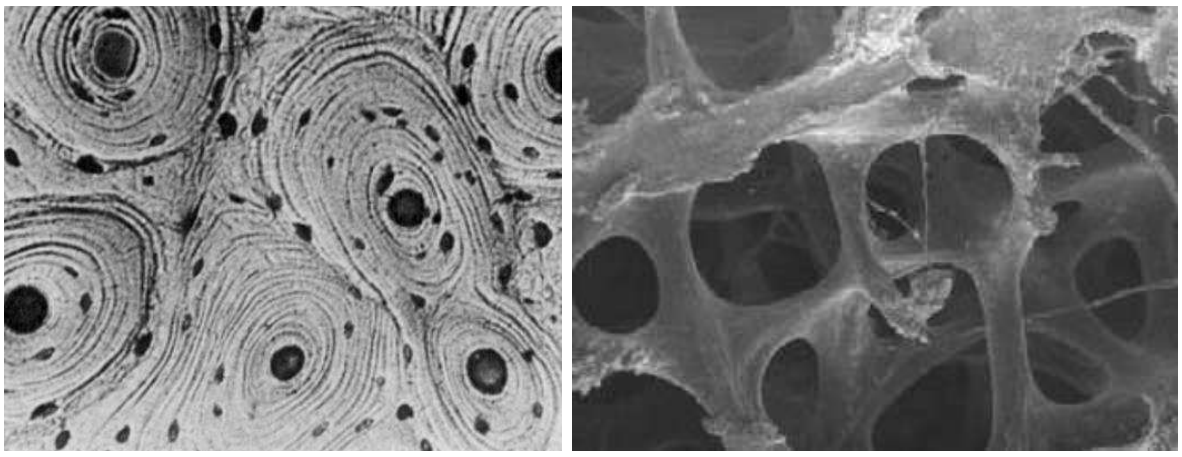
Compact (cortical) bone is the hard material that accounts for almost 80 percent of the total bone mass of an adult skeleton and forms a protective outer shell. It consists of layers of bone (lamellae), which are organized around central canals in which blood vessels, nerves, connective tissue and lymphatic vessels are found. A cortical bone structure on a morphological level is



(a) Bone section [7] (b) A close view [8]

Figure 2.1: Compact and cancellous bone structure

depicted in Figure 2.2(a). Cancellous (trabecular) bone is the porous material inside the compact bone. It accounts for 20% of the total bone mass, but 80% of the total bone surface. Cancellous bone is regarded as a three-dimensional inter-connected network of trabecular rods and plates [6], a morphological view is shown in Figure 2.2(b).



(a) Cortical bone (b) Trabecular bone

Figure 2.2: Morphological views of cortical and trabecular bone structure [9]

Biologically, the compact and cancellous bone tissues are very similar in composition; the difference lies in the arrangement of the microstructure. The classification of bone tissue as compact or cancellous is based on the relative density (Section 2.2.2); i.e. the ratio of specimen density to that of fully dense cortical bone [6].

2.2 Introduction to bone biomechanics

This subsection addresses the bone's biomechanical properties which includes: bone density, bone adaption and remodeling, and mechanical properties of bone tissues. As the most commonly used non-invasive way of obtaining bone density information, the CT imaging is firstly introduced.

2.2.1 CT imaging

Medical imaging is a technique to produce images of human body for clinical purpose or medical science. It provides a fast and non-invasive means to acquire information of human tissue and organs. The first imaging technique available in modern medicine is the X-ray radiography. During the scanning, an X-ray generator produces a beam of X-rays which is projected towards an object. These X-rays penetrate the object and are absorbed according to the density and composition of the different areas of the object. The X-rays passing through the object are captured by either a film or a digital detector which produces a 2D representation of all the structures of an object superimposed on each other [10]. Images produced with radiography have several limitations, e.g. only 2D images are produced and no information in the slice thickness direction is available, and the image can be blurred due to the superposition of different structures in the thickness direction.

To obtain 3D images of an object, advanced imaging techniques have been developed, such as computed tomography (CT), magnetic resonance imaging (MRI), ultrasound, etc. These techniques are all based on tomography which is the process of generating 2D sectional images through a 3D object by using any kind of penetrating wave. The modern tomography involves gathering data from multiple directions and feeding the data into a tomographic reconstruction software algorithm processed by a computer which constructs a series of cross-sectional scans [11]. The image produced in a tomography is a tomogram. In the current study, the main research focus is concentrated on images acquired by computed tomography using X-ray. Note that although the original idea of computed tomography is closely related to X-ray, other penetrating waves, such as neutron wave is also incorporated in the modern CT.

CT − short for computed tomography, is a powerful nondestructive imaging technique for generating a series of cross-sectional 2D images inside one object. Each 2D CT image is composed of a square image matrix. Each pixel is a square picture element that makes up the matrix [12]. Since a CT section has a finite thickness, each pixel actually represents a small volume element, or voxel [13]. Every voxel has been traversed during the scan by numerous X-ray photons and the intensity of the transmitted radiation is measured by detectors. From the radiation intensity information, the density or attenuation value of the tissue at each point in the slice can be calculated. Specific attenuation values are assigned to each individual voxel. These values are compared with the attenuation value of water and displayed on a scale of arbitrary units named Hounsfield units (HUs) [14]. Normally the voxels are represented as 12-bit binary numbers, and therefore have $2^{12} = 4096$ possible values. These values are arranged on a scale from -1024 HU to +3071 HU, calibrated so that -1024 HU is the attenuation produced by air and 0 HU is the attenuation produced by water. Tissue and bone then produce attenuations in the positive range. The reading in Hounsfield units is also called the

CT number [15].

2.2.2 Bone density

The bone density, also called "apparent density", is defined as dry, fat-free bone mass per unit bulk volume [16]. It can be obtained by straightforward measurements of the weight and volume of an excised bone specimen. The apparent density of human bones varies from $0.05 g/cm^3$ [17] (loose trabecular bone) to $1.9 g/cm^3$ (dense cortical bone) [18]. Another density term which can also be straightforwardly measured is the so called ash density, which is the density of the inorganic material alone. For this purpose, a bone specimen is firstly washed to remove bone marrow and afterwards ashed in an oven for a certain time period, e.g., in [19] at 650°C for 24 hours, or in two steps 100°C for 24 hours and 600°C for another 24 hours [20]. Afterwards the ash density is calculated by ash weight per unit bulk volume[1].

Generally, bone density can be obtained through invasive measurements on the bone specimen in laboratory, or through non-invasive measuring techniques like Dual Energy X-ray Absorptiometry (DEXA) or quantitative computed tomography (QCT). The DEXA is an enhanced form of X-ray technology used to generate a 2D calibrated digitized radiograph of a bone. The grey level of the pixels indicates the amount of X-rays passing through the bone. Often employed to diagnose osteoporosis, the DEXA measures not the bone apparent density, but the areal bone mineral content, so called the bone mineral density (BMD). The QCT is referred to as a dedicated computed tomography technique used to quantify some property of the tissue, e.g., bone mineral density, lung nodule calcification, body fat measurement, etc. Being one of the most precise technologies to measure the BMD of cross-sectional bone images, it enables a sensitive determination of local changes [6]. It has been shown in [21] that trabecular bone mineral density measured by QCT and the apparent density measured directly are significantly positively correlated. In general, QCT is performed on standard clinical CT scanners. During the scanning, a calibration phantom which is comprised of several chambers containing different concentrations of K_2HPO_4 is placed as close to the bone as possible to minimize errors introduced by non-uniformity of the CT numbers within the scan field [22]. As a good linear relationship exists between the CT number and the corresponding K_2HPO_4 concentration [23], a new density term, the so called "equivalent mineral density", denoted as ρ_{EQM} with unit (mg/cm^3), is often employed to substitute the HU values in terms of the linear relation.

2.2.3 Bone adaption and remodeling

Bone is an adaptive structure in which bone cells sense mechanical loading and adapt bone mass and structure accordingly [6]. Wolff's law [24] states that bone's structures are developed such that they can resist forces acting upon them in the most suited manner. They adapt both their external conformation and their internal architecture to changes in external loading conditions. Bone's adaption to external loading is considered as an ongoing and life-long process, which consists of two subprocesses: resorption and formation in which bone material

[1] The bulk volume is defined as the volume per unit mass of a dry material after ashing plus the volume of the air between its particles

is broken down or newly generated respectively. The formation process takes place when local stresses are intensified, resulting in a rise in bone density on the microscopic scale and an increment in bone external dimensions on a macroscopic scale. When local stresses are lowered, the resorption process is triggered, causing a lowering of the density on the microscopic scale and a decrease in bone external dimensions on a macroscopic scale.

2.2.4 Mechanical properties of bone tissues

Human bone can, by reasonable approximation, be regarded as a linear elastic material in the range of regular physical loading. Similar to many structural materials for instance metal, there is no significant difference in behavior between tension and compression within the range of small deformation [25]. Generally the analysis of the material properties of metal is under the homogeneous and isotropic assumption, namely, the metal is assumed to be uniform in composition and have the same mechanical behavior in all directions. Human bone, however, is neither a homogeneous nor an isotropic material, but rather heterogeneous and anisotropic, meaning that its material composition differs from place to place and its mechanical properties vary according to the direction.

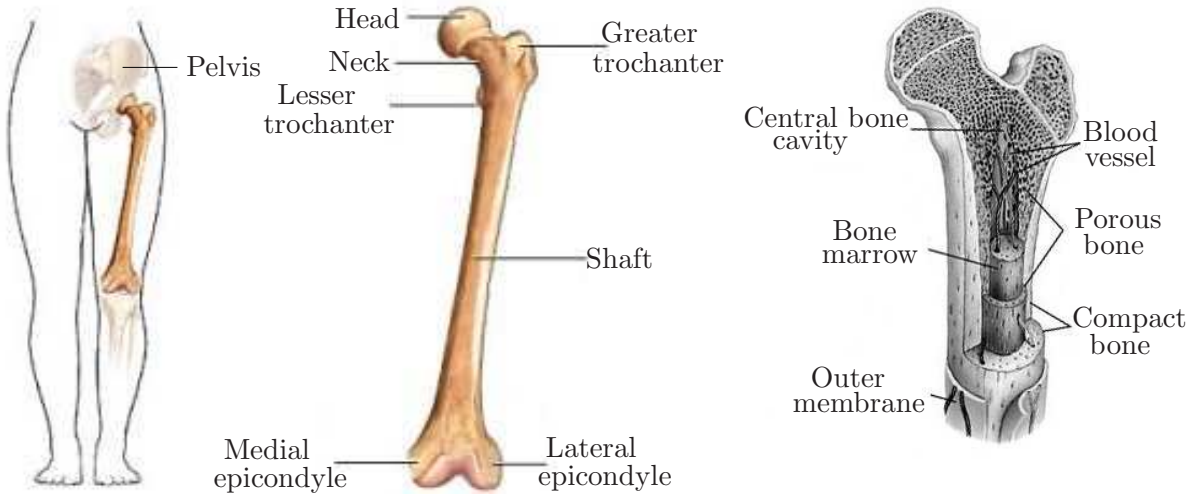
The main reason for the bone's heterogeneity and anisotropy is its structure, which has an irregular, yet optimized, arrangement and orientation of the components. The collagen fibres, lamellae, laminae and blood vessels show a clear tendency to be oriented along the length of a long bone [26]. Consequently the different structures of cortical bone and trabecular bone result in different mechanical properties, as concluded by Rice *et al.*. Trabecular and cortical bone should, therefore, be regarded as different engineering materials [27]. Usually the trabecular bone is considered to be an orthotropic material with different properties or strengths in three different orthogonal directions. Among the three directions, there is one principle direction in which the trabecular structure behaves stiffer than in the other two directions [28]. The cortical bone is treated as a transversely isotropic material, which has different material properties in one principle direction that is perpendicular to the plane of isotropy. Orthotropy or transversely isotropy is a special case of anisotropy with reduced number of independent material constants resulted from symmetry. As a reasonable approximation, isotropic material models are often used to simplify the biomechanical analysis [29, 30].

The bone is a highly heterogeneous material, in which its density as well as its mechanical properties vary greatly at different location in the bone, even for the same bone type (trabecular or cortical) [31, 32]. Many studies have shown that the mechanical properties of the bone can be related to the bone apparent density [33, 34], or directly related to the bone equivalent mineral density ρ_{EQM} without computing the apparent density [35, 29, 36]. In these studies the bone mechanical properties are statistically related to the bone density with various approximated correlations, among which power-law relations are often assumed. These relations are found to be highly variable in terms of species and anatomical site [37, 38, 39]. For a specific bone type and anatomical site, experiments have been carried out on bone specimens to obtain the relations. However, deterministic values are difficult to acquire due to the fact that different relations are reported when different experimental methods are used, e.g., compression test, ultrasonic methods, or FE analysis of microscopic bone structure [37, 34, 33, 28]. Note that even when the same method is used, the relations may be influenced by the geometries of

specimen [25]. Other parameters, such as age, sex and liquid content, are also reported to have an influence on the material properties of the bone.

2.3 Introduction to femur

The femur (thigh bone), located in the upper part of the human leg, is one of the two strongest bones in our body. In the stance posture, the femur is not vertical but inclining gradually downward, as indicated in Figure 2.3(a). The inclination angle is approximately 7° and larger in females than males.



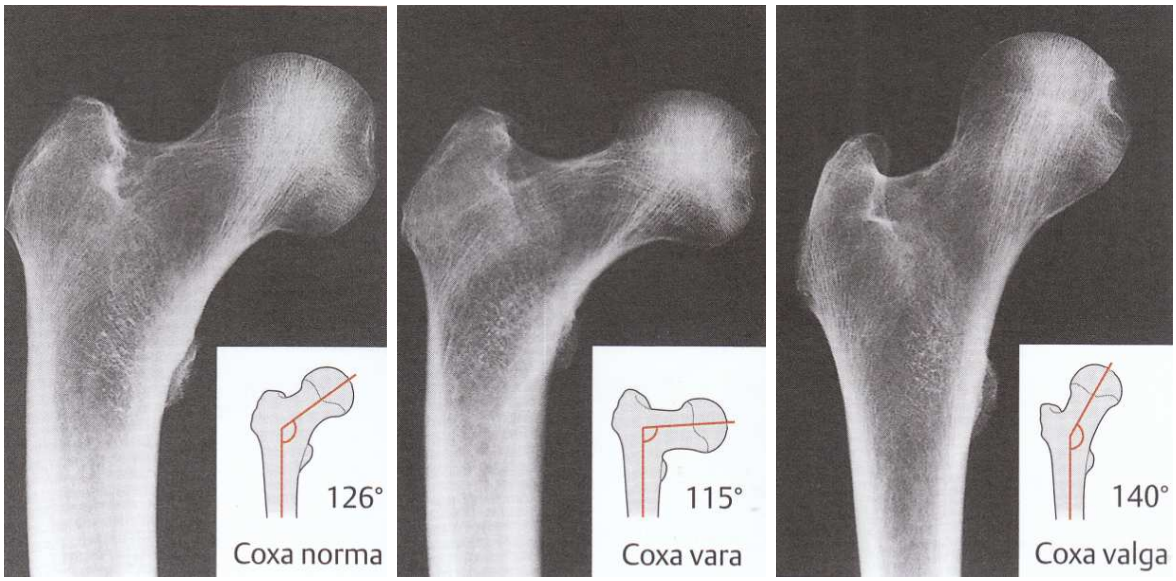
(a) A complete view [40] (b) A longitudinal section view [41]

Figure 2.3: Femur osteology

The upper extremity of the femur, which consists of a head, a neck, a greater and a lesser trochanter, is called the proximal femur. The femur's head is spherical and forms more than a hemisphere that articulates with the pelvis. Its surface is smooth and coated with a smooth layer called articular cartilage, except for a small roughened pit below and behind the center of the head. The femur's neck, which connects the head with the femur shaft, forms a wide angle of about 125° with the shaft in adults. This angle differs in person and gender (see Figure 2.4 for various angles of femur from different adults).

This wide angle enables human to swing the limb from the pelvis easily. Geometrically the neck is narrow in the middle while wider at its lateral than median end. The two trochanters, greater and lesser, are eminences that work as leverage to the muscles and enable the axial rotation of the femur. The greater trochanter is large, quadrilateral and located at the joint of the neck to the shaft. The lesser trochanter is conical and situated at the lower and back end of the neck.

The femur shaft has a cylindrical shape, but is slightly convex in its front and concaved at its back. It has a hollow structure with a longitudinal ring-like wall mainly composed of compact



(a) (b) (c)

Figure 2.4: Three radiographs of femur with different head angles [42]

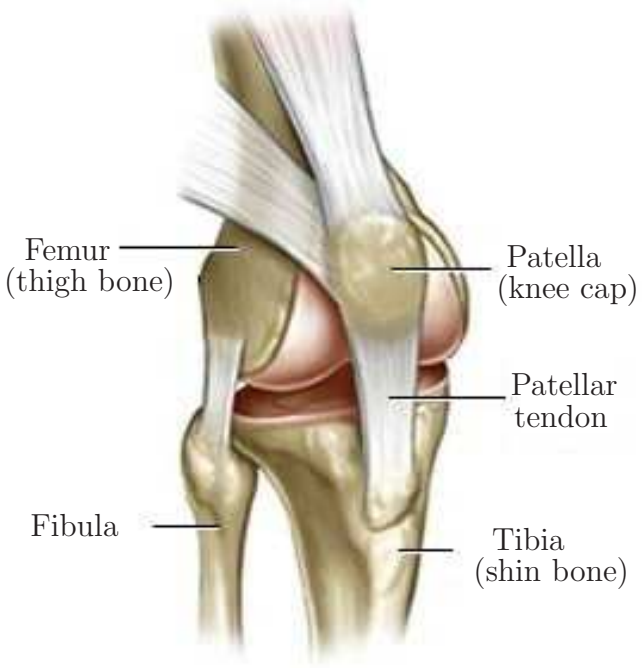


Figure 2.5: A schematic view of the knee-joint [43]

bone and a large medullary cavity in the middle. The lower extremity of the femur, which is larger than the upper extremity, has a cuboidal form. It forms the knee-joint together with the upper end of the tibia, the upper end of the fibula and the kneecap, see Figure 2.5.

In each part of the femur, the internal structures of the bone are aligned to form a pattern which is adapted to the mechanical requirement caused by the load transmitted from the

femur's head [44]. Throughout the femur, the bone adapts itself so efficiently that the bony material is arranged in the paths of the maximum internal stresses, thereby its inner structure is nearly optimum for an economical load transmission from the head to the tibia [45]. Researchers have performed already in the first half of the last century computations of strains and stresses to quantify the mechanical load transfer within a bone [44, 46, 47]. The relationship between the computed internal stresses resulting from the loading on the femur's head, and the inner structure of various sections of the femur is in good agreement with the theoretically ideal relationships existing between stress and inner structure for efficiency and maximum economy. Therefore, the following laws for bony structure are assumed to hold for the femur [44, 48]:

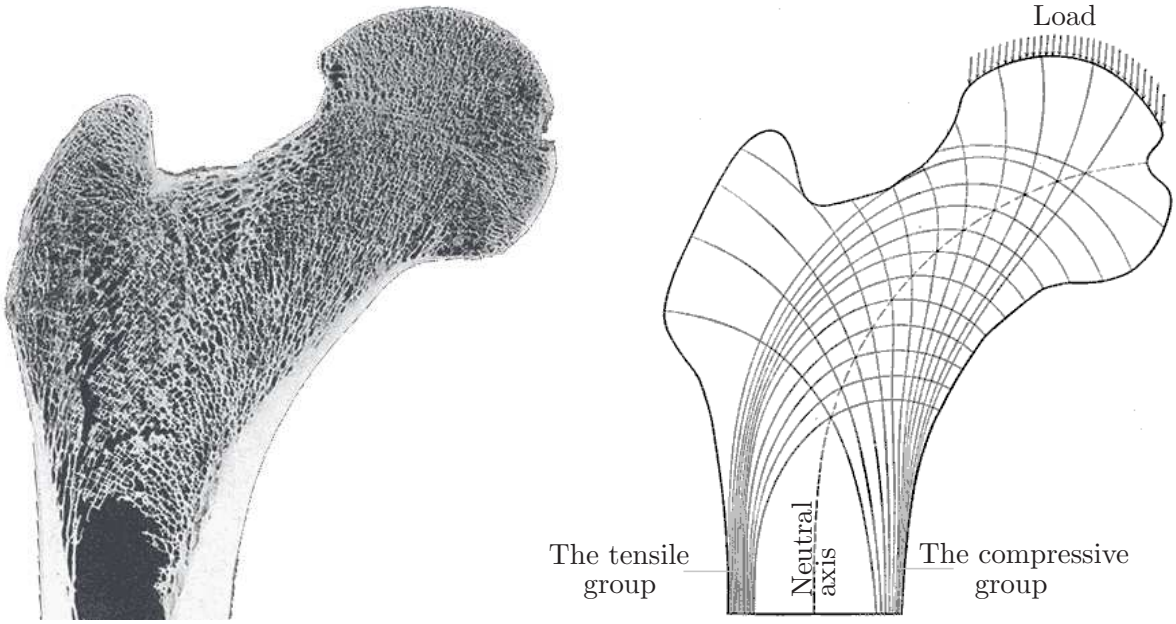
"1. The inner structure and external form of human bone are closely adapted to the mechanical conditions existing at every point in the bone."

"2. The inner architecture of normal bone is determined by definite and exact requirements of mathematical and mechanical laws to produce a maximum of strength with a minimum of material."

The inner architecture of the upper femur is depicted in Figure 2.6(a) and the principle stresses obtained by mathematical calculations are depicted in Figure 2.6(b). In the femur shaft, the cortical wall has its maximum thickness in the middle and becomes thinner as it reaches the articular surface at the upper part of the shaft, where cavity is replaced by cancellous bone gradually. As a result, at the proximal region of the femur the cancellous bone is enclosed by a thin compact layer. The upper femur consists mainly of cancellous bone which is composed of two systems of trabeculae, compressive and tensile. The compressive system of trabeculae originates from the medial part of the shaft and radiates in curves upwards in a fan-like style. The tensile system of the trabeculae originates from the lateral part of the shaft and spreads upwards in an arch-like style. These two systems intersect with each other orthogonally. The compressive system consists of two groups of trabeculae: the secondary compressive group and the principal compressive group; while the tensile system is comprised of three groups: the greater trochanter group, the principal tensile group and the secondary tensile group. A 2D schematic overview of the trabecular pattern of a human femur is shown in Figure 2.7.

1. The secondary compressive group:
This group of trabeculae originates from the inner boundary of the shaft near the lesser trochanter. After aligning along the curving shaft for about 5cm, these trabeculae start to separate smoothly with an angle of around 45° upwards and outwards to the region of the greater trochanter. The upper curved paths end in the region near the upper neck, while lower paths end in the region of the greater trochanter. Trabeculae in this group form a thin and porous structure with large spaces in between [49].

2. The principal compressive group:
This group of trabeculae originates from the medial part of the shaft and radiates upwards smoothly. They form paths reaching the upper part of the articular surface of the femur's head with a small curvature. Being regarded as a prolongation of the femur shaft, this group of trabeculae is much thicker and organizes itself in a much denser manner in contrast to the



(a) A cross-sectional view (b) Calculated principle stress directions

Figure 2.6: Proximal femur and the principle stress orientations [44]

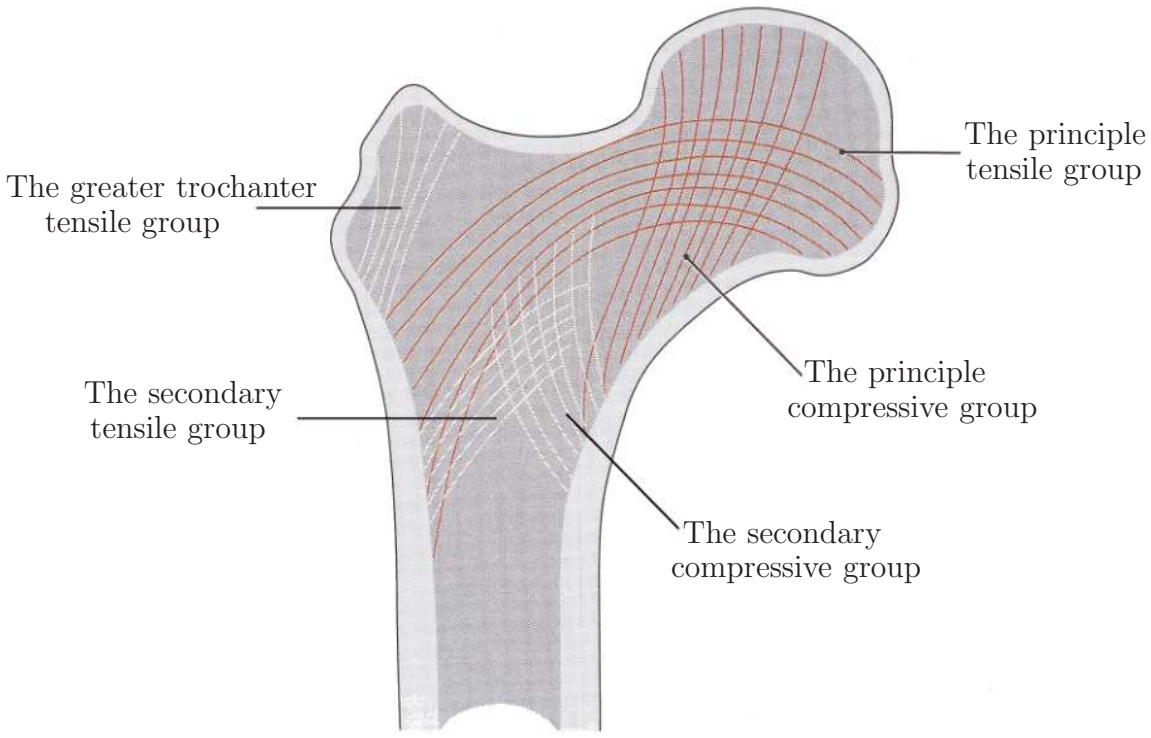


Figure 2.7: A schematic view of the trabecular pattern [42]

secondary group. These trabeculae paths are orthogonal to both the paths of the principle tensile group and the articular surface of the femur's head.

3. The greater trochanter tensile group:
Starting from the outer part of the shaft below the greater trochanter, trabeculae of the greater trochanter tensile group spring upwards crossing the greater trochanter region and end at the upper part of the greater trochanter. Some of them intersect with the paths of the secondary compressive group in right angels. The trabeculea of this group bear much less stresses which results in a slender structure.

4. The principal tensile group:
Originating from the outer part of the shaft, the principal tensile group of trabeculae radiates upwards and inwards across the femur neck and reaches the lower part of the head. Functioning as the main tension stress transmitter, these trabeculae are characterized in thinner shape but larger spacing in comparison with the ones of the principal compressive group.

5. The secondary tensile group:
The trabeculae of the secondary tensile group begin from the outer part of the shaft and expand in a region lower than the principal tensile group. They also radiate upwards and inwards and end either irregularly or at the medial part of the neck and shaft after crossing the neutral axis.

2.4 Femur mechanical analysis with the finite element method

The finite element method has been used in biomechanical simulations since the 70's. It provides a means to simulate a natural physiological/biomechanical phenomenon and to test various hypotheses on the simulated model under multiple conditions in order to deduce an approximate solution [50]. In biomechanics, FEM-simulation has become a well appreciated research tool for the prediction of stress responses. In the specific area of human femur, researchers have implemented various FE techniques to compute the femur's mechanical response and further to simulate human femur adaption, femoral head fracture and bone-prosthesis interaction, etc. [51, 52, 53, 54]. Reliability and efficiency studies on these methods have been conducted by many researchers through experimental validations [55, 1, 22].

Mainly differentiated by modeling and meshing, the FEA methods in Femur analysis can be categorized into two types: the "voxel based" method and the "structure based" method. This section will stress two methods in the FE analysis procedure of human femur that consists of four steps: modeling, mesh generation, material assignment, and solution.

2.4.1 FE Modeling

CT scanning is the most commonly used non-invasive approach to obtain femur models. CT data describe the scanned femur with a series of pictures, on which different volume units are assigned with proper HU values. To extract a parametric description of femur geometry, in the "voxel based" method the voxel data from a CT scan can be directly used after a simple step of thresholding and segmentation (if acquired); while in the "structure based"

method a geometric model which has a smooth surface description must be firstly constructed or imported from a surface generation program in a preprocessing step.

2.4.2 Mesh generation

The "voxel based" method directly converts the CT voxel data to 8-noded hexahedral elements, each enclosing a given number of CT voxels. In each element linear shape functions are commonly employed. This kind of method was initially proposed by Keyak [56] and has been widely adopted to produce FE models of microscopic structures, e.g., small substructures of trabecular bone [57, 58, 55].

Based on the geometric model obtained in the modeling step, in the "structure based" method a mesh with either hexahedral or tetrahedral elements can be generated. Linear or quadratic shape functions are often used in each element.

A comparative study on the two methods has been carried out in [55] to investigate the influence of different meshes on computational efficiency and accuracy. Three "structure based" meshes and one "voxel based" mesh were compared numerically based on the same CT-derived femur model: (a) a manually generated hexahedral mapped mesh, (b) an automatically generated tetrahedral mesh, (c) an automatically generated hexahedral mesh, and (d) an automatically generated voxel-based hexahedral mesh. The "voxel based" method has the advantage of easy mesh generation and shows good accuracy in displacements and interior stresses, while the meshes generated using "structure based" methods shows higher accuracy throughout the entire femur. The main reason for the errors in the "voxel based" method is due to the singularities occurring at the jagged boundaries of hexahedral meshes during FE analysis. On the contrary, the high accuracy in the "structure based" method is contributed by a more accurate geometric description and model discretization which entails higher computational cost.

An extension to the "structure based" method, which combines high order shape functions with a structure-based tetrahedral mesh has been implemented in [1]. Different from the general "structure based" method in which low order basis functions are used, only a relatively coarse mesh is necessary when high order basis functions are used.

2.4.3 Material assignment

For all types of meshes above, averaged material properties, averaged densities for instance, are assigned to each element. The assignment is straightforward in the "voxel based" method, since the mesh is directly related to the voxel data. However, in the "structure based" method, special assignment techniques are required since the elements are not predefined. In both methods, Young's modulus is assigned to every element based on the corresponding averaged density.

The solution step is the same as in the standard finite element methods and is therefore not further discussed.

Chapter 3

The p-version finite element method

3.1 Introduction

The finite element method (FEM) is one of the most powerful tools to obtain numerical solutions for partial differential equations that appear in various engineering fields, e.g., medical, civil and mechanical engineering. Generally, the finite element method helps to predict the behavior of a physical model with certain accuracy, which is restricted by various assumptions that are introduced to simplify the problem under certain consideration and the discretization error introduced by the finite element method itself. Both errors have to be controlled appropriately to ensure a reliable simulation.

The model error in finite element computations is due to the introduction of various physically motivated approximations or simplifications when deriving a specific mechanical model for a certain problem, e.g., using simplified constitutive law. The discretization error can be controlled by either applying the h-version or p-version refinement. In h-version [59, 60, 61] the mesh is locally or globally refined and low order basis functions (typically linear or quadratic) with a fixed polynomial degree are commonly employed. In p-version FEM [62] the finite element mesh is kept fixed and the polynomial degrees of the shape functions are increased to achieve convergence to the unknown solution of the underlying mathematical model. The mesh in p-version is generally coarse; however, for linear elliptic problems with smooth solutions it shows exponential rate of convergence in the energy norm. When combining the p-version with a proper local mesh refinement to an hp-version, exponential convergence rate can even be achieved for a class of non-smooth problems.

The aim of this chapter is to give an introduction to high order finite element methods in 3D. For this purpose, the upcoming sections are arranged as follow: Section 3.2 introduces some basic principles in three-dimensional linear elasticity theory. In Section 3.3 the principle of virtual work which is the basis of the finite element method is introduced. Afterwards, introduction to the finite element method with special emphasis on the p-version is given in Section 3.4.

3.2 Basic principles in three-dimensional linear elasticity

3.2.1 Equilibrium equation

The equilibrium conditions for a three-dimensional solid are obtained from inspection of an infinitesimal volume element with length dx, dy and dz (shown in Figure 3.1). The equilibrium conditions are

$$\begin{aligned} \frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z} + f_x &= 0 \\ \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{yz}}{\partial z} + f_y &= 0 \\ \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \sigma_z}{\partial z} + f_z &= 0 \end{aligned} \tag{3.1}$$

where σ_x, σ_y, σ_z denote the normal stresses, τ_{xy}, τ_{yz}, τ_{xz} denote the shear stresses, and $\mathbf{f} = \begin{bmatrix} f_x & f_y & f_z \end{bmatrix}^T$ is the volume load vector.

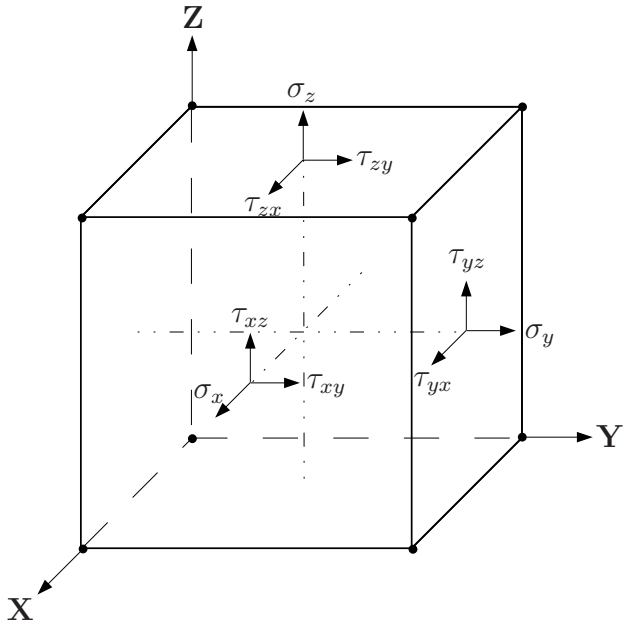


Figure 3.1: Stress components in 3D

3.2.2 Kinematics

The displacement $\mathbf{u}$ is described by the displacement vector

$$\mathbf{u} = \begin{bmatrix} u_x(x,y,z) & u_y(x,y,z) & u_z(x,y,z) \end{bmatrix}^T . \tag{3.2}$$

For small strains, the linear relation between strains and displacements is described by

$$\boldsymbol{\varepsilon} = \left[\begin{array}{cccccc} \varepsilon_x & \varepsilon_y & \varepsilon_z & \gamma_{xy} & \gamma_{yz} & \gamma_{zx} \end{array} \right]^T = \mathbf{L}\mathbf{u} \tag{3.3}$$

where

$$\begin{array}{ll} \varepsilon_x = \dfrac{\partial u_x}{\partial x}, & \gamma_{xy} = \dfrac{\partial u_x}{\partial y} + \dfrac{\partial u_y}{\partial x}, \\ \varepsilon_y = \dfrac{\partial u_y}{\partial y}, & \gamma_{yz} = \dfrac{\partial u_y}{\partial z} + \dfrac{\partial u_z}{\partial y}, \\ \varepsilon_z = \dfrac{\partial u_z}{\partial z}, & \gamma_{zx} = \dfrac{\partial u_z}{\partial x} + \dfrac{\partial u_x}{\partial z} \end{array} \tag{3.4}$$

and

$$\mathbf{L} = \left[\begin{array}{ccc} \dfrac{\partial}{\partial x} & 0 & 0 \\ 0 & \dfrac{\partial}{\partial y} & 0 \\ 0 & 0 & \dfrac{\partial}{\partial z} \\ \dfrac{\partial}{\partial y} & \dfrac{\partial}{\partial x} & 0 \\ 0 & \dfrac{\partial}{\partial z} & \dfrac{\partial}{\partial y} \\ \dfrac{\partial}{\partial z} & 0 & \dfrac{\partial}{\partial x} \end{array} \right] \tag{3.5}$$

is the linear strain operator.

3.2.3 Constitutive law

For three-dimensional isotropic linear elastic problems, the stress vector

$$\boldsymbol{\sigma} = \left[\begin{array}{cccccc} \sigma_x & \sigma_y & \sigma_z & \tau_{xy} & \tau_{yz} & \tau_{zx} \end{array} \right]^T \tag{3.6}$$

can be related to the strains by

$$\boldsymbol{\sigma} = \mathbf{C}\,\boldsymbol{\varepsilon} \tag{3.7}$$

where $\mathbf{C}$ is the linear elastic matrix.

For linear isotropic elastic material, $\mathbf{C}$ can be written as

$$\mathbf{C} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} (1-\nu) & \nu & \nu & 0 & 0 & 0 \\ \nu & (1-\nu) & \nu & 0 & 0 & 0 \\ \nu & \nu & (1-\nu) & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \tag{3.8}$$

according to the Hook's law, where E is the Young's modulus and ν is the Poisson's ratio.

3.2.4 Boundary condition

On the boundary $\partial\Omega = \Gamma_N \cup \Gamma_D$, $\Gamma_N \cap \Gamma_D = \emptyset$ either displacements or tractions can be defined as *boundary conditions*, in which prescribed displacements (*Dirichlet boundary conditions*) are given at Γ_D and prescribed tractions (*Neumann boundary conditions*) are given at Γ_N.

3.3 Weak formulation − Principle of virtual work

Weak formulations are an important tool in the analysis of mathematical equations which allow the transfer of problems in other fields such as partial differential equations into a set of integral equations. The main feature of weak formulations is that a partial differential equation is not required to hold pointwise yet only in a mean, integrated sense with respect to a set of well-defined test functions [63]. The establishment of the weak form can be applied to any equilibrium equation and is in solid mechanics mostly referred to as the *principle of virtual work* [64].

The *principle of virtual work* can be derived as follows.
Multiplying (3.1) by a *test function*

$$\mathbf{v} = \begin{bmatrix} v_x(x,y,z) \\ v_y(x,y,z) \\ v_z(x,y,z) \end{bmatrix} \quad \in \mathcal{V} \tag{3.9}$$

and integrating over the domain Ω yields

$$\int_\Omega \left[\left(\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z} \right) v_x + \left(\frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{yz}}{\partial z} \right) v_y + \left(\frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \sigma_z}{\partial z} \right) v_z \right] d\Omega + \int_\Omega \left(f_x v_x + f_y v_y + f_z v_z \right) d\Omega = 0 \tag{3.10}$$

where $\mathcal{V} = \{\mathbf{v}(\mathbf{x}) \in H^1(\Omega) : \mathbf{v} = \mathbf{0} \text{ on } \Gamma_D\}$ is a subspace of the Sobolev space $H^1(\Omega)$ [65] which consists of functions with square-integrable generalized derivatives.

The *test function* $\mathbf{v}$ can be regarded as a virtual displacement. Integrating by part (3.10) and applying the divergence theorem [66] yields

$$\int_{\Omega} \left(\sigma_x \varepsilon_x^{(v)} + \sigma_y \varepsilon_y^{(v)} + \sigma_z \varepsilon_z^{(v)} + \tau_{xy}\gamma_{xy}^{(v)} + \tau_{xz}\gamma_{xz}^{(v)} + \tau_{yz}\gamma_{yz}^{(v)}\right) d\Omega = \int_{\Gamma_N} \left(t_x v_x + t_y v_y + t_z v_z\right) d\Gamma + \int_{\Omega} \left(f_x v_x + f_y v_y + f_z v_z\right) d\Omega \tag{3.11}$$

in which $\varepsilon^{(v)}$ and $\gamma^{(v)}$ stand for virtual strains, t_x, t_y and t_z denote the tractions in X, Y and Z directions, v_x, v_y and v_z represent the virtual displacements in X, Y and Z directions.

Plugging in Equation (3.3) and (3.7), Equation (3.11) can be written as

$$\int_{\Omega} (\mathbf{L}\,\mathbf{v})^T\ \mathbf{C}\,(\mathbf{L}\,\mathbf{u})\,d\Omega = \int_{\Omega} \mathbf{v}^T\,\mathbf{f}\,d\Omega + \int_{\Gamma_N} \mathbf{v}^T\,\mathbf{t}\,d\Gamma\,. \tag{3.12}$$

where $\mathbf{L}$ is the linear strain operator in Equation (3.5). The left-hand side of Equation (3.12)

$$\mathcal{B}(\mathbf{u}\,,\mathbf{v}) := \int_{\Omega} (\mathbf{L}\,\mathbf{v})^T\ \mathbf{C}\,(\mathbf{L}\,\mathbf{u})\,d\Omega \tag{3.13}$$

is a bilinear form, which represents the virtual work of internal stresses.

The right-hand side of Equation (3.12)

$$\mathcal{F}(\mathbf{v}) := \int_{\Omega} \mathbf{v}^T\,\mathbf{f}\,d\Omega + \int_{\Gamma_N} \mathbf{v}^T\,\mathbf{t}\,d\Gamma \tag{3.14}$$

is a linear functional that expresses the virtual work of external stresses.

Equation (3.12) then reads

" *Find* $\mathbf{u}_{EX} \in \mathcal{S} = \{\mathbf{u}(\mathbf{x}) \in H^1(\Omega) : \mathbf{u} = \widehat{\mathbf{u}} \text{ on } \Gamma_D\}$, *such that*

$$\mathcal{B}(\mathbf{u}_{EX}\,,\mathbf{v}) := \mathcal{F}(\mathbf{v}) \quad \forall\, \mathbf{v} \in \mathcal{V} = \{\mathbf{v}(\mathbf{x}) \in H^1(\Omega) : \mathbf{v} = \mathbf{0} \text{ on } \Gamma_D\}\ " \tag{3.15}$$

where $\mathbf{u}_{EX}$ is the exact solution and $\widehat{\mathbf{u}}$ stands for prescribed displacements.

The strain energy is given by

$$\mathcal{U}(\mathbf{u}) := \frac{1}{2}\mathcal{B}(\mathbf{u},\mathbf{u}) \tag{3.16}$$

and the energy norm is defined as

$$\|\mathbf{u}\|_{E(\Omega)} = \sqrt{\mathcal{U}(\mathbf{u})} = \sqrt{\frac{1}{2}\mathcal{B}(\mathbf{u},\mathbf{u})}. \tag{3.17}$$

3.4 The finite element approximation

In general, the exact solution of (3.15) can only be obtained for problems with simple geometries and boundary conditions. For more complicated problems, numerical methods, e.g. the finite element method, are used to obtain approximate solutions to the weak form. Primarily, the finite element method requires a problem defined in geometric domain to be divided into a finite number of smaller elements. Each element is unique and may have different shapes, i.e. quadrilaterals or triangles in 2D and hexahedra or tetrahedra in 3D. Over each element, the unknown variables (e.g. displacements, strains, stresses, etc.) are approximated using either linear or higher-order polynomial expansions which depend on the geometrical locations used to define the element shape. The weak form is then discretized in a finite dimensional space. The discretized weak form can be expressed as a linear system which is obtained in terms of unknown parameters over each element. Linear algebra techniques are used to solve these equations [67].

3.4.1 Spatial discretization by the FEM

Using the FE approximate solution, denoted by $\mathbf{u}_{FE}$, to replace the exact solution $\mathbf{u}_{EX}$, the principle of virtual work (3.15) is rewritten as

"Find $\mathbf{u}_{FE} \in \mathcal{S}^h$, *such that*

$$\mathcal{B}(\mathbf{u}_{FE},\mathbf{v}) := \mathcal{F}(\mathbf{v}) \quad \forall\, \mathbf{v} \in \mathcal{V}^h \text{ "} \tag{3.18}$$

where $\mathcal{S}^h \subset \mathcal{S} = \{\mathbf{u}(\mathbf{x}) \in H^1(\Omega) : \mathbf{u} = \widehat{\mathbf{u}} \text{ on } \Gamma_D\}$ and $\mathcal{V}^h \subset \mathcal{V} = \{\mathbf{v}(\mathbf{x}) \in H^1(\Omega) : \mathbf{v} = \mathbf{0} \text{ on } \Gamma_D\}$ are the finite element subspaces.

In order to construct the finite subspace $\mathcal{S}^h$, the reference domain $\mathbf{\Omega}_0$ is subdivided into n_e non-overlapping subdomains, so called finite elements – $\mathbf{\Omega}^e$. From this, it follows

$$\bigcup_{e=1}^{n_e} \mathbf{\Omega}^e \approx \mathbf{\Omega}_0 \tag{3.19}$$

and

$$\mathbf{\Omega}^{e_i} \cap \mathbf{\Omega}^{e_j} = \emptyset \text{ for } e_i \neq e_j\,. \tag{3.20}$$

(3.20) also requires that each two subdomains touch with each other only at nodes, full edges or full faces to avoid hanging nodes [68].

The subspace $\mathcal{S}$ can be constructed using basis functions with local supports on the element $\mathbf{\Omega}^e$. The exact solution $\mathbf{u}_{EX}$ is approximated with the FE solution constructed by a linear combination of element shape functions $\mathbf{N}_i$.

$$\mathbf{u}_{EX} \approx \mathbf{u}_{FE} = \sum_{i=1}^{n_{modes}} \mathbf{N}_i \, \mathbf{u}_i \tag{3.21}$$

where $\mathbf{u}_i$ are the coefficients corresponding to $\mathbf{N}_i$ which are defined on the elements Ω^e fulfilling the requirement that all shape functions of an element are zero outside the element and its direct neighbors. Sorting $\mathbf{u}_i$ corresponding to element e into one element displacement vector $\mathbf{U}^e$ gives

$$\mathbf{u}^e_{FE} = \mathbf{N}^e \, \mathbf{U}^e \, . \tag{3.22}$$

The global displacement function $\mathbf{u}$ is obtained by assembly of all element displacement functions given as

$$\mathbf{u}_{FE} = \mathbf{N} \, \mathbf{U} \, . \tag{3.23}$$

Shape functions $\mathbf{N}$ are defined on a standard element and mapped to the actual element for numerical evaluations. The choice of shape functions and the mapping concept differentiate the h- and p-version finite element method as well as related variants. Different shape functions can be employed in the p-version FEM, one of them are the hierarchic shape functions.

3.4.2 Hierarchic shape functions for high-order finite elements

For constructing high-order basis functions, Szabó and Babuška [62] propose a hierarchical basis in which lower order shape functions are included in the set of higher order shape functions. The construction of this basis is based on orthogonal Legendre polynomials. The main difference of the hierarchic shape functions to non-hierarchic shape functions, e.g. shape functions constructed by Lagrange polynomials, is illustrated and compared. The following sections 3.4.2.1 and 3.4.2.2 follow closely the description of hierarchic shape function in [69].

3.4.2.1 Hierarchic shape function for one-dimensional problems

As a starting point for comparison, the standard finite element basis − nodal basis on a standard element $\Omega_{\text{st}} = (-1, 1)$ is firstly introduced, see the left-hand side of Table 3.1.

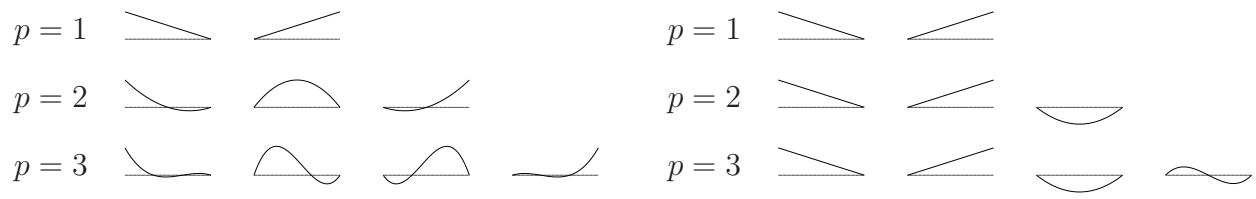


Table 3.1: Set of one-dimensional standard and hierarchic shape functions for $p = 1, 2, 3$

The standard finite element shape functions in one dimension are given by the set of Lagrange polynomials

$$N_i^p(\xi) = \prod_{j=1,\, j\neq i}^{p+1} \frac{\xi - \xi_j}{\xi_i - \xi_j}\,. \tag{3.24}$$

The points ξ_j where

$$N_i^p(\xi_j) = \delta_{ij} \tag{3.25}$$

are called nodes. Usually, the nodes are chosen to be equally distributed, i.e.

$$\xi_j = -1 + 2\,\frac{j-1}{p}, \quad j = 1, ..., p+1\,. \tag{3.26}$$

For each polynomial degree p a separate set of shape functions has to be defined, for example, for $p = 1$

$$\begin{aligned} N_1^1(\xi) &= 1/2(1-\xi) \\ N_2^1(\xi) &= 1/2(1+\xi) \end{aligned} \tag{3.27}$$

for $p = 2$

$$\begin{aligned} N_1^2(\xi) &= 1/2\,\xi\,(\xi-1) \\ N_2^2(\xi) &= (1+\xi)\,(1-\xi) \\ N_3^2(\xi) &= 1/2\,(\xi+1)\,\xi \end{aligned} \tag{3.28}$$

for $p = 3$

$$\begin{aligned} N_1^3(\xi) &= -1/16\,(3\,\xi+1)\,(3\,\xi-1)\,(\xi-1) \\ N_2^3(\xi) &= 9/16\,(\xi+1)\,(3\,\xi-1)\,(\xi-1) \\ N_3^3(\xi) &= -9/16\,(\xi+1)\,(3\,\xi+1)\,(\xi-1) \\ N_4^3(\xi) &= 1/16\,(\xi+1)\,(3\,\xi+1)\,(3\,\xi-1) \end{aligned} \tag{3.29}$$

etc. Note that the sum of all Lagrange polynomials for a given polynomial degree p equals unity

$$\sum_{i=1}^{p+1} N_i^p(\xi) \;=\; 1\,. \tag{3.30}$$

The space representable by the standard basis can also be represented by a hierarchical basis, see the right-hand side of Table 3.1. Note that the set of higher order basis functions includes all lower order shape functions. The set of one-dimensional hierarchic shape functions, introduced by Szabó and Babuška [62] is given by

$$N_1(\xi) \;=\; 1/2(1-\xi) \tag{3.31}$$
$$N_2(\xi) \;=\; 1/2(1+\xi) \tag{3.32}$$
$$N_i(\xi) \;=\; \phi_{i-1}(\xi), \quad i = 3,4,...,p+1 \tag{3.33}$$

with

$$\phi_j(\xi) \;=\; \sqrt{\frac{2j-1}{2}} \int_{-1}^{\xi} L_{j-1}(x)\,dx = \frac{1}{\sqrt{4j-2}}(L_j(\xi) - L_{j-2}(\xi))\,, \quad j = 2,3,... \tag{3.34}$$

where $L_j(\xi)$ are the Legendre polynomials. The linear functions $N_1(\xi), N_2(\xi)$ are called *nodal shape functions* or *nodal modes*. Because the functions $N_i(\xi)$, $i = 3,4,...$ vanish at the domain boundary

$$N_i(-1) \;=\; N_i(1) = 0, \quad i = 3,4,...\,, \tag{3.35}$$

they are called *internal shape functions*, *internal modes* or *bubble modes*. The orthogonality property of Legendre polynomials implies

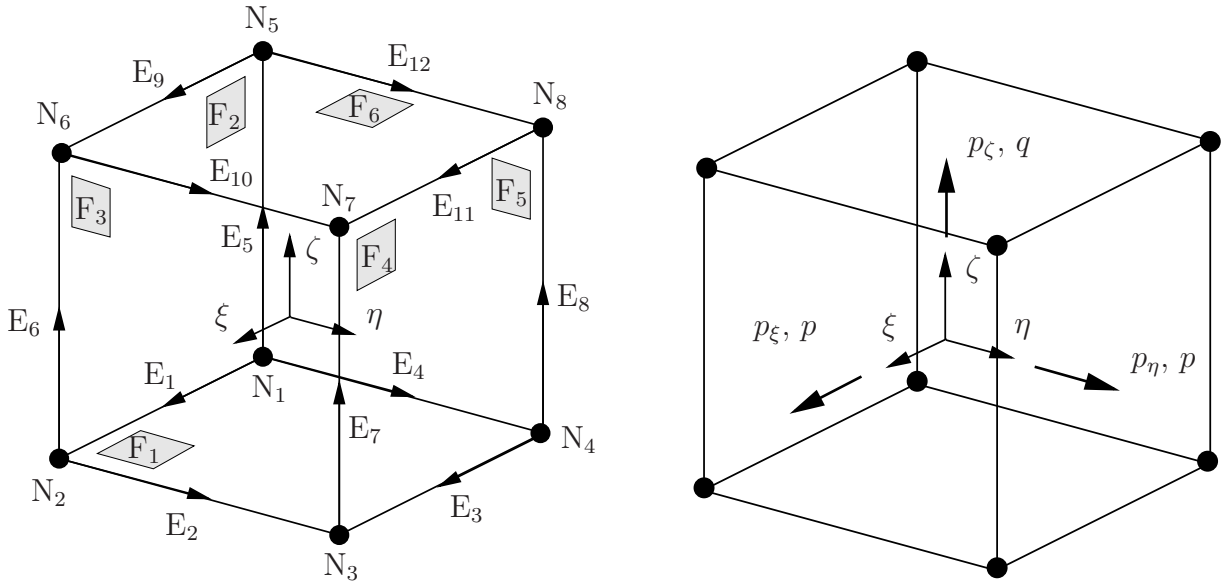
$$\int_{-1}^{1} \frac{d\,N_i}{d\,\xi}\,\frac{d\,N_j}{d\,\xi}\,d\,\xi = \delta_{ij}, \quad i \geq 3 \text{ and } j \geq 1 \quad \text{or} \quad i \geq 1 \text{ and } j \geq 3. \tag{3.36}$$

The construction of shape functions for two and three-dimensional Ansatz spaces can be easily done by simply forming the tensor product of one-dimensional hierarchic shape functions. In the next section the three-dimensional shape functions are presented for hexahedral elements.

3.4.2.2 Hierarchic shape function for three-dimensional problems

Using the basis functions introduced by SZABÓ and BABUŠKA [62], the p-version in three-dimensions is implemented based on a hexahedral element formulation. Hexahedral elements (see Figure 3.2) have some advantages, when being compared to their corresponding tetrahedral and pentahedral element formulations.

- Hexahedral element formulations lead to higher accuracy for low p.
- Hexahedral elements are especially well suited for thin-walled structures. One local variable can be identified to correspond to the thickness direction. Therefore it is possible to choose the polynomial degree in thickness direction differently from those in in-plane direction.
- The numerical integration of hexahedral elements can be readily performed using a Gaussian quadrature scheme.



$\Omega^{\mathrm{h}}_{\mathrm{st}} = [(-1,1) \times (-1,1) \times (-1,1)]$ $\quad$ $\mathcal{S}^{p_\xi,p_\eta,p_\zeta}_{\mathrm{ts}}(\Omega^{\mathrm{h}}_{\mathrm{st}}), \mathcal{S}^{p_\xi,p_\eta,p_\zeta}_{\mathrm{ps}}(\Omega^{\mathrm{h}}_{\mathrm{st}}), \mathcal{S}^{p,p,q}(\Omega^{\mathrm{h}}_{\mathrm{st}})$

Figure 3.2: Standard hexahedral element $\Omega^{\mathrm{h}}_{\mathrm{st}}$: definition of nodes, edges, faces and polynomial degree

The three-dimensional shape functions can be classified into four groups:

1. **Nodal modes:** The nodal modes

$$N^{\mathrm{N}_i}_{1,1,1}(\xi,\eta,\zeta) \;=\; \frac{1}{8}(1+\xi_i\,\xi)(1+\eta_i\,\eta)(1+\zeta_i\zeta), \quad i=1,...,8 \tag{3.37}$$

   are the standard trilinear shape functions, well known from the isoparametric eight-noded brick element. (ξ_i,η_i,ζ_i) are the local coordinates of the i-th node.

2. **Edge modes:** These modes are defined separately for each individual edge. If we consider, for example, edge E_1 (see Figure 3.2), the corresponding edge modes read:

$$N^{\mathrm{E}_1}_{i,1,1}(\xi,\eta,\zeta) \;=\; \frac{1}{4}(1-\eta)(1-\zeta)\phi_i(\xi) \tag{3.38}$$

3. **Face modes:** These modes are defined separately for each individual face. If we consider, for example, face F_1, the corresponding face modes read:

$$N^{\mathrm{F}_1}_{i,j,1}(\xi,\eta,\zeta) \;=\; \frac{1}{2}(1-\zeta)\phi_i(\xi)\phi_j(\eta) \tag{3.39}$$

4. **Internal modes:** The internal modes

$$N^{\mathrm{int}}_{i,j,k}(\xi,\eta,\zeta) \;=\; \phi_i(\xi)\phi_j(\eta)\phi_k(\zeta) \tag{3.40}$$

   are purely local and vanish at the faces of the hexahedral element.

The indices i,j,k of the shape functions denote the polynomial degrees in the local directions ξ,η,ζ.

Three different types of basis spaces have been implemented: the *trunk space* $\mathcal{S}^{p_\xi,p_\eta,p_\zeta}_{\mathrm{ts}}(\Omega^{\mathrm{h}}_{\mathrm{st}})$, the *tensor product space* $\mathcal{S}^{p_\xi,p_\eta,p_\zeta}_{\mathrm{ps}}(\Omega^{\mathrm{h}}_{\mathrm{st}})$ and the *anisotropic tensor product space* $\mathcal{S}^{p,p,q}(\Omega^{\mathrm{h}}_{\mathrm{st}})$. A detailed description of the three spaces can be found in DÜSTER [70, 69]. For the definition of the spaces $\mathcal{S}^{p_\xi,p_\eta,p_\zeta}_{\mathrm{ts}}(\Omega^{\mathrm{h}}_{\mathrm{st}})$ and $\mathcal{S}^{p,p,q}(\Omega^{\mathrm{h}}_{\mathrm{st}})$ see also SZABÓ and BABUŠKA [62].

The polynomial degree for the Ansatz spaces $\mathcal{S}^{p_\xi,p_\eta,p_\zeta}_{\mathrm{ts}}(\Omega^{\mathrm{h}}_{\mathrm{st}})$ and $\mathcal{S}^{p_\xi,p_\eta,p_\zeta}_{\mathrm{ps}}(\Omega^{\mathrm{h}}_{\mathrm{st}})$ can be varied separately in each local direction (see Figure 3.2). The difference between the trunk space and the tensor product space is relevant for the face modes and the internal modes. For explanation, we first consider the face modes, for example the modes for face 1. Indices i,j denote the polynomial degrees of the face modes in ξ and η direction, respectively.

Face modes (face F_1): $N^{\mathbf{F}_1}_{i,j,1}(\xi,\eta,\zeta) = \frac{1}{2}(1-\zeta)\phi_i(\xi)\phi_j(\eta)$

trunk space	**tensor product space**
$i = 2,...,p_\xi - 2$	$i = 2,...,p_\xi$
$j = 2,...,p_\eta - 2$	$j = 2,...,p_\eta$
$i + j = 4,...,\max\{p_\xi, p_\eta\}$	

The definition of the set of internal modes is very similar. Indices i,j,k now denote the polynomial degrees in the three local directions ξ,η and ζ.

internal modes: $N^{\mathbf{int}}_{i,j,k}(\xi,\eta,\zeta) = \phi_i(\xi)\phi_j(\eta)\phi_k(\zeta)$

trunk space	tensor product space
$i = 2, ..., p_\xi - 4$	$i = 2, ..., p_\xi$
$j = 2, ..., p_\eta - 4$	$j = 2, ..., p_\eta$
$k = 2, ..., p_\zeta - 4$	$k = 2, ..., p_\zeta$
$i + j + k = 6, ..., \max\{p_\xi, p_\eta, p_\zeta\}$	

3.4.3 Equation system derived from the weak form

Starting from the weak form

$$\mathcal{B}(\mathbf{u}_{FE}\,, \mathbf{v}) \;=\; \mathcal{F}(\mathbf{v})\,, \tag{3.41}$$

rewriting the virtual displacement

$$\mathbf{v} = \mathbf{N}\,\mathbf{V} \tag{3.42}$$

and plugging in Equation (3.23) and (3.42) into (3.41) yields

$$\mathcal{B}(\mathbf{N}\,\mathbf{U}\,, \mathbf{N}\,\mathbf{V}) \;=\; \mathcal{F}(\mathbf{N}\,\mathbf{V})\,. \tag{3.43}$$

Plugging in (3.13) and (3.14) into (3.43), Equation (3.43) then reads

$$\mathbf{V}^T \Big(\int_\Omega (\mathbf{L}\,\mathbf{N})^T\;\mathbf{C}\,(\mathbf{L}\,\mathbf{N})\,d\Omega\Big)\,\mathbf{U} = \mathbf{V}^T \int_\Omega \mathbf{N}^T\,\mathbf{f}\,d\Omega \;+\; \mathbf{V}^T \int_\Gamma \mathbf{N}^T\,\mathbf{t}\,d\Gamma\,. \tag{3.44}$$

As this statement is valid for any value of the virtual displacement, it follows

$$\int_\Omega (\mathbf{L}\,\mathbf{N})^T\;\mathbf{C}\,(\mathbf{L}\,\mathbf{N})\,d\Omega\,\mathbf{U} = \int_\Omega \mathbf{N}^T\,\mathbf{f}\,d\Omega \;+\; \int_\Gamma \mathbf{N}^T\,\mathbf{t}\,d\Gamma \tag{3.45}$$

Defining

$$\mathbf{K} = \int_\Omega (\mathbf{L}\,\mathbf{N})^T\;\mathbf{C}\,(\mathbf{L}\,\mathbf{N})\,d\Omega\,,\; \mathbf{F} = \int_\Omega \mathbf{N}^T\,\mathbf{f}\,d\Omega \;+\; \int_\Gamma \mathbf{N}^T\,\mathbf{t}\,d\Gamma\,, \tag{3.46}$$

Equation (3.45) is transformed to

$$\mathbf{K}\,\mathbf{U} = \mathbf{F} \tag{3.47}$$

where $\mathbf{K}$ is the global stiffness matrix and $\mathbf{F}$ is the global load vector.

Having discretized the problem domain into finite elements and element-wise shape functions $\mathbf{N}$, the global stiffness matrix $\mathbf{K}$ and global load vector $\mathbf{F}$ are computed by assembly of all element stiffness matrices and element load vectors respectively:

$$\mathbf{K} = \mathop{\mathsf{A}}_{e=1}^{n_e} \mathbf{K}^e \,, \qquad \mathbf{F} = \mathop{\mathsf{A}}_{e=1}^{n_e} \mathbf{F}^e \tag{3.48}$$

The element stiffness matrix and the load vector can be computed by

$$\mathbf{K}^e = \int_\Omega (\mathbf{L}\,\mathbf{N})^T \; \mathbf{C}^e \, (\mathbf{L}\,\mathbf{N}) \, dx \, dy \, dz \,, \tag{3.49}$$

$$\mathbf{F}^e = \int_\Omega \mathbf{N}^T \, \mathbf{f} \, d\Omega + \int_\Gamma \mathbf{N}^T \, \mathbf{t} \, d\Gamma \,. \tag{3.50}$$

When computing the element stiffness matrix $\mathbf{K}^e$, the computation of derivatives of the shape functions is involved. Since the shape functions are defined in the local coordinate system, a mapping of a general eight-node element in the global coordinate system to the standard element $[[-1,\,1] \times [-1,\,1] \times [-1,\,1]]$ with local coordinates $\boldsymbol{\xi} = (\,\xi,\,\eta,\,\zeta\,)^T$ is required. For instance, computation of the first derivative of the shape function N_i with respect to global coordinate system involves the chain rule

$$\begin{bmatrix} \dfrac{\partial N_i}{\partial \xi} \\ \dfrac{\partial N_i}{\partial \eta} \\ \dfrac{\partial N_i}{\partial \zeta} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} & \dfrac{\partial z}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} & \dfrac{\partial z}{\partial \eta} \\ \dfrac{\partial x}{\partial \zeta} & \dfrac{\partial y}{\partial \zeta} & \dfrac{\partial z}{\partial \zeta} \end{bmatrix} \begin{bmatrix} \dfrac{\partial N_i}{\partial x} \\ \dfrac{\partial N_i}{\partial y} \\ \dfrac{\partial N_i}{\partial z} \end{bmatrix} = \mathbf{J} \begin{bmatrix} \dfrac{\partial N_i}{\partial x} \\ \dfrac{\partial N_i}{\partial y} \\ \dfrac{\partial N_i}{\partial z} \end{bmatrix} \tag{3.51}$$

in which $\mathbf{J}$ is the Jacobian matrix. In order to obtain derivatives with respect to the global coordinates, one has to invert the Jacobian matrix:

$$\begin{bmatrix} \dfrac{\partial N_i}{\partial x} \\ \dfrac{\partial N_i}{\partial y} \\ \dfrac{\partial N_i}{\partial z} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \dfrac{\partial N_i}{\partial \xi} \\ \dfrac{\partial N_i}{\partial \eta} \\ \dfrac{\partial N_i}{\partial \zeta} \end{bmatrix} \tag{3.52}$$

The Jacobian matrix must be regular (i.e., its determinant $det\mathbf{J} \neq 0$) in order to compute its inverse.

The integration over one element can be computed by

$$\int_{\Omega^e} (\cdot)\, d\Omega = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} (\cdot) \det \mathbf{J}\,(\xi,\eta,\zeta)\; d\xi\, d\eta\, d\zeta \tag{3.53}$$

and the element stiffness matrix can be computed in the local coordinate system as:

$$\mathbf{K}^e = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} (\mathbf{L}\,\mathbf{N})^T\; \mathbf{C}^e\,(\mathbf{L}\,\mathbf{N})\, det\mathbf{J}\, d\xi\, d\eta\, d\zeta \tag{3.54}$$

where **LN** is also denoted as the standard strain-displacement matrix **B** which will be used hereafter.

Chapter 4

The finite cell method

The finite cell method is a fictitious domain method combined with high-order shape functions. The basic idea is to extend the partial differential equation beyond the physical domain, up to the boundaries of a fictitious domain which is rectangular. In the new domain the exact solution is approximated by high-order polynomials.

4.1 Basic formulation

Let us assume that on a domain Ω with the boundary $\partial\Omega$ a problem of linear elasticity is described in weak form by

$$\mathcal{B}(\mathbf{u}, \mathbf{v}) = \mathcal{F}(\mathbf{v}) \tag{4.1}$$

where the bilinear form is

$$\mathcal{B}(\mathbf{u}, \mathbf{v}) := \int_{\Omega} [\mathbf{L}\,\mathbf{v}]^T \mathbf{C}\,[\mathbf{L}\,\mathbf{u}]\, d\Omega\,. \tag{4.2}$$

The domain of computation is now embedded in the domain Ω_e with the boundary $\partial\Omega_e$, see Figure 4.1.

Without losing generality, the homogeneous Dirichlet boundary condition is assumed to be applied along Γ_D, while the Neumann boundary condition is defined along Γ_N, where $\partial\Omega = \Gamma_D \cup \Gamma_N$, $\Gamma_D \cap \Gamma_N = \emptyset$ and $\Gamma_N = \Gamma_N^I \cup \Gamma_N^H$. Γ_N^H and Γ_N^I denotes the homogeneous and inhomogeneous Neumann boundary condition, respectively. The linear functional of the weak form (4.1) is

$$\mathcal{F}(\mathbf{v}) := \int_{\Omega} \mathbf{v}^T \mathbf{f}\, d\Omega + \int_{\Gamma_N} \mathbf{v}^T \mathbf{t}_N\, d\Gamma \tag{4.3}$$

where $\mathbf{f}$ are volume loads and $\mathbf{t}_N$ are prescribed tractions.

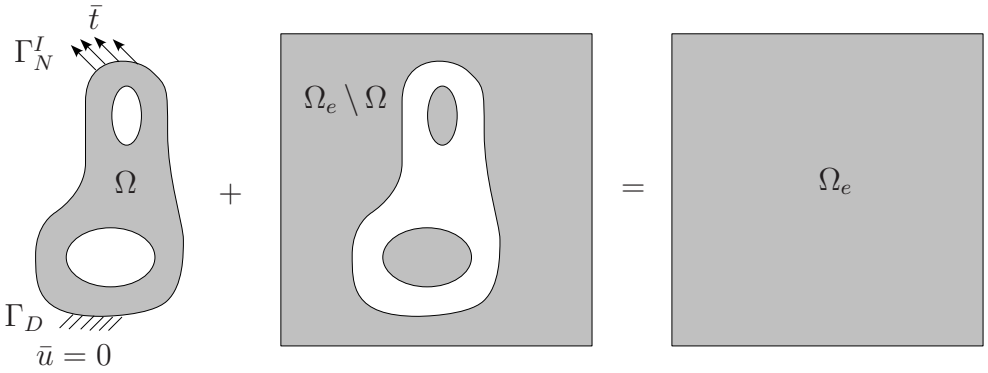


Figure 4.1: The domain Ω is embedded in Ω_e

The original physical domain Ω is now embedded in the domain Ω_e with the boundary $\partial\Omega_e$, see Figure 4.1 in which the situation of boundary conditions is also depicted for simplicity. The interface between Ω and the extended domain $(\Omega_e \setminus \Omega)$ is defined as $\Gamma_I = \partial\Omega \setminus (\partial\Omega \cap \partial\Omega_e)$. Following [71], the displacement variable $\mathbf{u}$ is extended as:

$$\mathbf{u} = \begin{cases} \mathbf{u}^1 & \text{in } \Omega \\ \mathbf{u}^2 & \text{in } \Omega_e \setminus \Omega \end{cases} \tag{4.4}$$

while the transition conditions guarantee continuity at the interface between Ω and $\Omega_e \setminus \Omega$:

$$\begin{aligned} \mathbf{t}^1 &= \mathbf{t}^2 \qquad \text{on } \Gamma_I \\ \mathbf{u}^1 &= \mathbf{u}^2 \qquad \text{on } \Gamma_I \end{aligned} \tag{4.5}$$

Boundary conditions are set for $\partial\Omega_e$

$$\begin{aligned} \bar{\mathbf{t}} &= \mathbf{0} \qquad \text{on } \Gamma_{e,N}\,. \\ \bar{\mathbf{u}} &= \mathbf{0} \qquad \text{on } \Gamma_{e,D} \end{aligned} \tag{4.6}$$

where $\Gamma_{e,N}$ and $\Gamma_{e,D}$ are the Neumann and Dirichlet boundaries of Ω_e respectively, $\partial\Omega_e = \Gamma_{e,N} \cup \Gamma_{e,D}$, and $\Gamma_{e,N} \cap \Gamma_{e,D} = \emptyset$.

The weak form of the equilibrium equation for the embedding domain Ω_e is given as

$$\mathcal{B}_e\left(\mathbf{u}\,,\mathbf{v}\right) \;=\; \mathcal{F}_e(\mathbf{v}) \tag{4.7}$$

where the bilinear form is

$$\mathcal{B}_e(\mathbf{u}\,,\mathbf{v}) := \int_{\Omega_e} \left[\mathbf{L}\,\mathbf{v}\right]^T \mathbf{C}_e \left[\mathbf{L}\,\mathbf{u}\right] d\Omega \tag{4.8}$$

in which $\mathbf{C}_e$ is the elasticity matrix of the embedding domain, given as

$$\mathbf{C}_e = \alpha\, \mathbf{C}\,, \tag{4.9}$$

where

$$\alpha = \begin{cases} 1.0 & \text{in } \Omega \\ 0.0 & \text{in } \Omega_e \setminus \Omega \end{cases} \tag{4.10}$$

Plugging in Equation (4.9) and (4.10) into (4.8), the bilinear functional turns to

$$\begin{aligned} \mathcal{B}_e(\mathbf{u}\,,\mathbf{v}) \;&:=\; \int_{\Omega} [\mathbf{L}\,\mathbf{v}]^T\, \mathbf{C}\, [\mathbf{L}\,\mathbf{u}]\, d\Omega + \int_{\Omega_e \setminus \Omega} [\mathbf{L}\,\mathbf{v}]^T\, \mathbf{0}\, [\mathbf{L}\,\mathbf{u}]\, d\Omega \\ &=\; \int_{\Omega_e} [\mathbf{L}\,\mathbf{v}]^T\, \alpha\mathbf{C}\, [\mathbf{L}\,\mathbf{u}]\, d\Omega := \mathcal{B}\,(\mathbf{u}\,,\mathbf{v})\,. \end{aligned} \tag{4.11}$$

The linear functional

$$\mathcal{F}_e(\mathbf{v}) := \int_{\Omega_e} \mathbf{v}^T\, \alpha\, \mathbf{f}\, d\Omega + \int_{\Gamma_N} \mathbf{v}^T\, \bar{\mathbf{t}}\, d\Gamma + \int_{\Gamma_{e,N}} \mathbf{v}^T\, \bar{\mathbf{t}}\, d\Gamma \tag{4.12}$$

includes volume loads $\mathbf{f}$, prescribed traction $\bar{\mathbf{t}}$ along Γ_N interior to Ω_e and prescribed traction at the boundary of the embedding domain. The last term can be assumed zero due to Equation (4.6).

The embedding domain is now discretized in a mesh that is independent of the original domain. These new elements, which are differentiated from classical elements, are called finite cells. As a simple and beneficial initialization, cells are assumed to be rectangular hexahedra which results in constant Jacobian matrices for cell-wise mappings, see Figure 4.2.
The union of all cells forms the extended domain

$$\Omega_e = \bigcup_{c=1}^{n_c} \Omega^c \tag{4.13}$$

where Ω^c is the domain represented by a cell, and n_c is total number of cells resulting from division of the embedding domain. At the discretized level, the bilinear form (4.8) turns to

$$\mathcal{B}_e(\mathbf{u}\,,\mathbf{v}) := \sum_{c=1}^{n_c} \int_{\Omega^c} [\mathbf{L}\,\mathbf{v}]^T\, \alpha\, \mathbf{C}\, [\mathbf{L}\,\mathbf{u}]\, d\Omega\,. \tag{4.14}$$

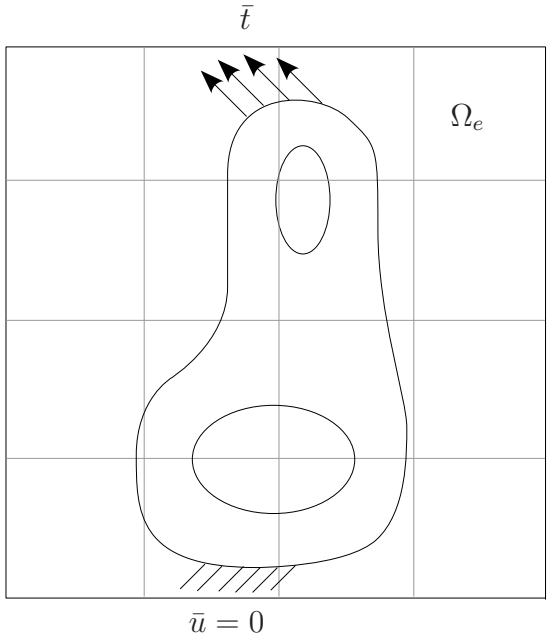


Figure 4.2: Discretization of the embedding domain using rectangular cells

The displacement variable in each cell is approximated as

$$\mathbf{u} = \mathbf{N}\,\mathbf{U} \tag{4.15}$$

where $\mathbf{N}$ denotes the matrix of shape functions and $\mathbf{U}$ is the vector of unknowns. In the current implementation, Legendre polynomial based hierarchical shape function [62, 70, 72] is used, see Section 3.4.2. This implementation allows an h-extension for low and high polynomial orders by refinement of cells as well as a p-extension on a fixed mesh of (coarse) cells. Based on the Bubnov-Galerkin approach, $\mathbf{v} = \mathbf{NV}$, plugging (4.15) into (4.14) the finite cell formulation is given as

$$\mathbf{K}\,\mathbf{U} = \mathbf{F} \tag{4.16}$$

where $\mathbf{K}$ is the global stiffness matrix and $\mathbf{F}$ is the global load vector. They are obtained by assembling cell matrices and cell load vectors respectively.

4.2 Boundary conditions

In FCM, application of boundary conditions on surfaces which do not conform to the mesh embedding the original domain Ω is not as straightforward as it is in the standard FEM. Special methods are needed to handle Neumann and Dirichlet boundary conditions separately.

4.2.1 Neumann boundary conditions

Homogeneous Neumann boundary conditions are firstly considered. This "zero traction condition" is equivalent to assuming material with zero stiffness in the extended domain. Inhomogeneous boundary conditions can be imposed by explicitly including the second term in Equation (4.12), i.e. by integrating over the boundary Γ_N lying in the interior of Ω_e, see Figure 4.3 for two-dimensional case. A detailed description of load vector computation for the three-dimensional case is given in Section 4.3.2.

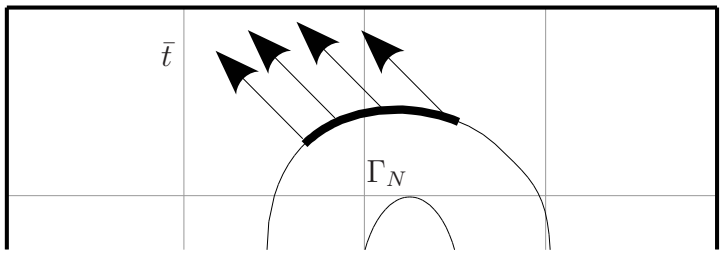


Figure 4.3: The Neumann boundary conditions

4.2.2 Dirichlet boundary conditions

In [2, 3] a "stiff strip" of material from Γ_D up to the Dirichlet boundary $\Gamma_{e,D}$ of the extended domain is assumed to apply homogeneous boundary conditions, see the dark region highlighted in Figure 4.4. In this way, instead of being applied directly on the physical boundary, Dirichlet boundary conditions can be applied on the cell boundary. This strip is generally assumed to be of one magnitude stiffer than in the original computational domain to produce reasonably accurate results.

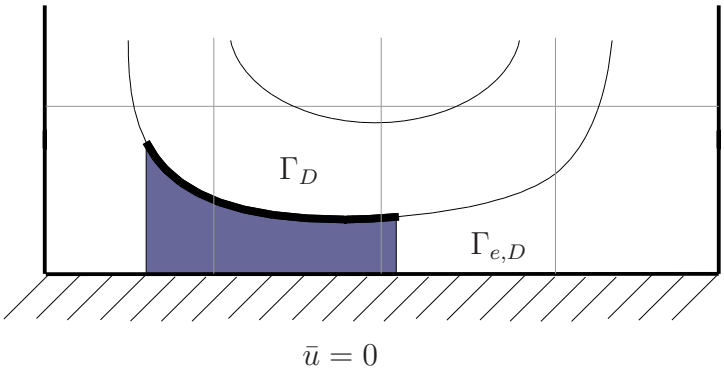


Figure 4.4: The Dirichlet boundary conditions

Inhomogeneous boundary condition can be applied with a natural extension of the "stiff strip" approach. In the literature several ways to impose inhomogeneous Dirichlet boundary conditions have been proposed, for instance, penalization or Lagrangian methods. In [2, 3, 73]

the penalization method and the Nitsche method are implemented to prescribe either the inhomogeneous or homogeneous displacement only at the boundary Γ_D.

4.3 Numerical integration

4.3.1 Computation of cell stiffness matrices

The approximation of the original problem (4.2) over the domain has been replaced by a problem over an embedding domain, yet with discontinuous coefficients. Therefore, the integrand in (4.14) may be discontinuous within cells being cut by the boundary $\partial\Omega$. An adaptive integration is necessary to capture this discontinuity. Different integration schemes have been investigated. Low order integration like trapezoidal rule on a refined grid of sub-cells can be used while the integration points are distributed uniformly in the cell. Another alternative is to use a quad tree technique subdividing the cut cell to non-uniform smaller cells for adaptive integration [74].

In [3] a composed scheme which can integrate over either uniformly or adaptively divided sub-cells has been proven to be a reliable method with high approximation accuracy. This implementation is based on a hexahedral element applying hierarchic high-order shape functions.

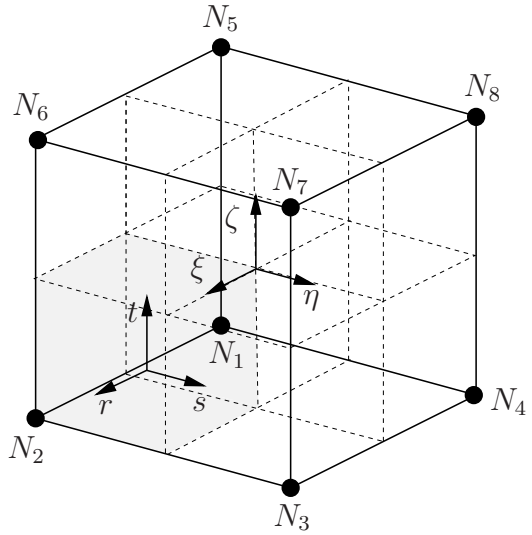


Figure 4.5: Composed integration of hexahedral elements based on sub-cells [3]

The mapping from the standard element to the global coordinate system is defined as

$$\mathbf{x} \;=\; \mathbf{Q}^c(\xi,\eta,\zeta) = \sum_{i=1}^{8} N_i(\xi,\eta,\zeta)\,\mathbf{X}_i \tag{4.17}$$

where $\mathbf{X}_i = (X_i, Y_i, Z_i)^T$ denote the global coordinates of the eight nodes and

$$N_i(\xi,\eta,\zeta) \;=\; \frac{1}{8}\,(1+\xi_i\,\xi)(1+\eta_i\,\eta)(1+\zeta_i\zeta), \quad i=1,...,8 \tag{4.18}$$

are the tri-linear shape functions, where (ξ_i,η_i,ζ_i) are the local coordinates of the ith node. The embedding domain is discretized with rectangular hexahedra. So the mapping reduces to

$$\mathbf{x} \;=\; \mathbf{Q}^c(\xi,\eta,\zeta) = \begin{bmatrix} X_1+\frac{1}{2}(1+\xi)h_x \\ Y_1+\frac{1}{2}(1+\eta)h_y \\ Z_1+\frac{1}{2}(1+\zeta)h_z \end{bmatrix} \tag{4.19}$$

which results in a constant Jacobian matrix

$$\mathbf{J}^c \;=\; \frac{1}{2}\begin{bmatrix} h_x & 0 & 0 \\ 0 & h_y & 0 \\ 0 & 0 & h_z \end{bmatrix} \tag{4.20}$$

where h_x, h_y and h_z denote the cell size with respect to the x, y and z direction, respectively. With a constant Jacobian matrix, the hierarchic shape functions defined on the standard element which are mapped on the global coordinate system remain polynomials.

A composed integration is applied to compute the stiffness matrix

$$\mathbf{K}^c \;=\; \int_\zeta \int_\eta \int_\xi (\mathbf{B}^c)^T(\boldsymbol{\xi})\,\alpha(\mathbf{x}(\boldsymbol{\xi}))\,\mathbf{C}\,\mathbf{B}^c(\boldsymbol{\xi})\,\mathrm{det}\mathbf{J}^c\,d\xi d\eta d\zeta \tag{4.21}$$

where the cell to be integrated is divided into n_{sc} *sub-cells*, see Figure 4.5. These sub-cells are used for integration purposes only. In order to establish a relation between the coordinates of the block-shaped sub-cell $\mathbf{r} = (r,s,t)^T$ and the cell, a linear mapping function

$$\boldsymbol{\xi} \;=\; \tilde{\mathbf{Q}}_c^{sc}(r,s,t) = \begin{bmatrix} \xi_1+\frac{1}{2}(1+r)h_\xi \\ \eta_1+\frac{1}{2}(1+s)h_\eta \\ \zeta_1+\frac{1}{2}(1+t)h_\zeta \end{bmatrix} \tag{4.22}$$

is applied, where (ξ_1,η_1,ζ_1) are the local coordinates of the anchor point of the sub-cell and h_ξ, h_η, h_ζ denote the size of the sub-cell. Since the sub-cells are block-shaped, the mapping results in a constant Jacobian matrix

$$\tilde{\mathbf{J}}_c^{sc} \;=\; \mathrm{grad}^T\tilde{\mathbf{Q}}_c^{sc}(r,s,t) = \frac{1}{2}\begin{bmatrix} h_\xi & 0 & 0 \\ 0 & h_\eta & 0 \\ 0 & 0 & h_\zeta \end{bmatrix}. \tag{4.23}$$

The stiffness matrix of cell c is obtained by carrying out the composed integration over the n_{sc} sub-cells

$$\mathbf{K}^c = \sum_{sc=1}^{n_{sc}} \int_t \int_s \int_r (\mathbf{B}^c)^T(\boldsymbol{\xi}(\mathbf{r}))\, \alpha(\mathbf{x}(\boldsymbol{\xi}(\mathbf{r})))\, \mathbf{C}\, \mathbf{B}^c(\boldsymbol{\xi}(\mathbf{r}))\, \mathrm{det}\mathbf{J}^c\, \mathrm{det}\tilde{\mathbf{J}}_c^{sc}\, drdsdt\,. \tag{4.24}$$

In (4.24) it is also necessary to account for the determinant of the Jacobian matrix $\tilde{\mathbf{J}}_c^{sc}$ which is due to the change of variables. The shape functions are the same as in (4.21); however, the mapping $\boldsymbol{\xi}(\mathbf{r}) = \tilde{\mathbf{Q}}_c^{sc}(r,s,t)$ has to be applied to establish the relation between the coordinates of the sub-cell (r,s,t) and the cell (ξ,η,ζ). The sub-cells can be created so as to have a uniform spacing or an adaptive algorithm based on an octree data structure can be applied. The adaptive algorithm may be applied to automatically control the refinement of the sub-cells, so that in those regions where the integrand exhibits a strong variation (e.g. jumps due to the incorporation of the domain's boundary) a more accurate quadrature is performed. A different quadrature scheme may be chosen for each of the sub-cells.

To avoid ill conditioning, cells completely outside Ω can be ignored for integration and assembly, or a small non-zero α can be used for such cells. Numerical experiences show that any α as small as 10^{-10} can replace zero. The domain integral in the linear functional (4.12) can be dealt with in the extended domain by introducing α and using the same procedure as for the bilinear form.

4.3.2 Computation of cell load vectors

The cell load vector consists of two parts

$$\mathbf{F}^c = \mathbf{F}_f^c + \mathbf{F}_{\bar{t}}^c = \int_{\Omega_e} (\mathbf{N}^c)^T\, \alpha\, \mathbf{f}\, d\Omega + \int_{\Gamma_N^I} (\mathbf{N}^c)^T\, \bar{\mathbf{t}}\, d\Gamma \tag{4.25}$$

where $\mathbf{F}_f^c$ represents the volume load and $\mathbf{F}_{\bar{t}}^c$ results from the surface traction [3]. $\mathbf{F}_f^c$ can be computed through volume integration in a similar manner as the stiffness matrix. To account for inhomogeneous Neumann boundary conditions defined on Γ_N^I the boundary integral has to be computed cell-wise to determine the load vector $\mathbf{F}_{\bar{t}}^c$ due to a surface traction $\bar{\mathbf{t}}$ acting on Γ_N^I. Since the boundary Γ_N^I will normally not coincide with the boundary of the embedding domain Ω_e, the computation of the cell load vector cannot be obtained by simply integrating over the boundary of the cells, since Γ_N will generally intersect the cells.

In order to compute the load vector one needs to have a parametric description of the surface on which the traction is acting in order to perform the integration. Figure 4.6 depicts a triangulated surface intersecting a hexahedral element. The local coordinates of the triangular element is given as (u,v). The cell load vector based on a traction $\bar{\mathbf{t}}$ acting on a surface spanned by n_f^c facets embedded in cell c can be computed as

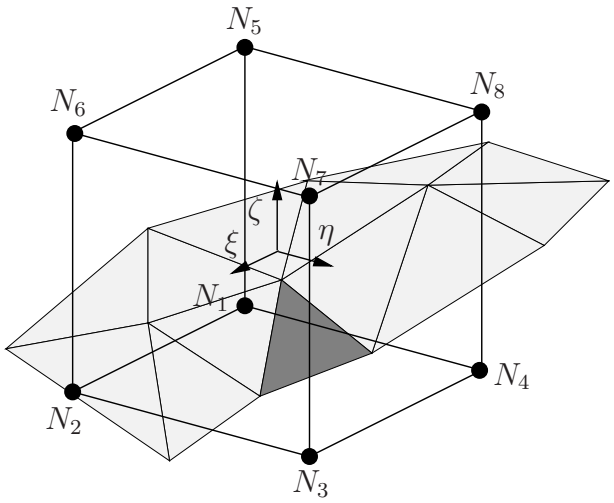


Figure 4.6: Computation of load vector by integrating over the Neumann boundary

$$\mathbf{F}_{\bar{t}}^{c} = \sum_{f=1}^{n_f^c} \int_v \int_u (\mathbf{N}^{\mathbf{c}})^T(\boldsymbol{\xi})\, \bar{\mathbf{t}}\, \det \mathbf{J}_t^f \, dudv\,. \tag{4.26}$$

where $\mathbf{J}_t^f$ denotes the Jacobian matrix which is a mapping from the global coordinates to the local coordinates of the triangular facet f. In this case, a two-dimensional surface (u, v) is embedded into a three-dimensional space (x, y, z). So the Jacobian matrix can be computed as the norm of the vector normal to the surface. The vector normal is the cross product of the two vectors $\mathbf{x}_u^f = \dfrac{\partial \mathbf{x}^f}{\partial u}$ and $\mathbf{x}_v^f = \dfrac{\partial \mathbf{x}^f}{\partial v}$.

Accordingly $\mathbf{J}_t^f$ can be written as

$$\mathbf{J}_t^f = |\mathbf{x}_u^f(u,v) \times \mathbf{x}_v^f(u,v)| \tag{4.27}$$

where $\mathbf{x}^f$ denotes the global coordinates of the triangular facet.

Equation (4.26) then reads

$$\mathbf{F}_{\bar{t}}^{c} = \sum_{f=1}^{n_f^c} \int_v \int_u (\mathbf{N}^{\mathbf{c}})^T(\boldsymbol{\xi})\, \bar{\mathbf{t}}\, |\mathbf{x}_u^f(u,v) \times \mathbf{x}_v^f(u,v)|\, dudv \tag{4.28}$$

In order to compute $\mathbf{x}^f$, a mapping from the local coordinates (u, v) of the standard triangular element to the global Cartesian coordinates is required. Afterwards, the integration can be

computed with a two-dimensional quadrature rule defined on a standard triangular element. Note that $\mathbf{N^c}$ denotes the shape functions which are high order polynomials in the FCM. To accurately integrate the product of the shape functions and the traction vector over the area a high order integration rule is required. However, some high order quadrature rules for triangular elements published in finite element textbooks and papers contain either negative weights or points outside of the integral domain [75, 76, 77]. To ensure an accurate computation, a mapping from the local coordinates (u, v) defined on a standard quadrilateral element ($[-1,-1]\times[1,1]$) to the global coordinates (x, y, z) of the triangle is used instead, see Equation (4.29).

$$\mathbf{x}^f = \mathbf{Q}^f(u,v) = \sum_{i=1}^{4} N_i(u,v)\mathbf{X}_i^f \tag{4.29}$$

$\mathbf{X}_i^f = (X_i^f, Y_i^f, Z_i^f)^T$ denote the global coordinates of the three nodes of the triangle. By coalescing two adjacent vertices of the quadrilateral element, the quadrilateral standard domain can be transformed to fit the triangular facet, see Figure 4.7 where $\mathbf{X}_4^f = \mathbf{X}_3^f$ has been chosen.

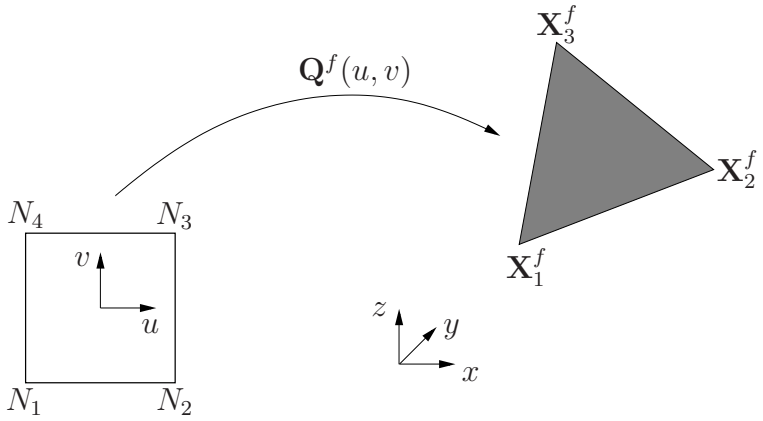


Figure 4.7: Parameterization of facets [3]

In Equation (4.29)

$$N_i(u,v) = \frac{1}{4}(1+u_iu)(1+v_iv) \tag{4.30}$$

are the bi-linear shape functions, where (u_i, v_i) denote the local coordinates of the ith node of the standard quadrilateral. Applying the mapping function (4.29) makes it possible to perform the integration with a two-dimensional Gaussian quadrature defined on a standard quadrilateral. It should be borne in mind that the process of triangulating a curved surface introduces a discretization error. Commercial CAD tools are designed to control this discretization error, i.e. the deviation from the (true) curved surface, by specifying the maximum chordal deviation tolerance of the facets.

As can be seen from Figure 4.6 it is very likely that the Gaussian points related to one facet fall within different cells (hexahedra). It is accordingly important to ensure that, during the integration over each facet, the contribution of this integration point refers to the cell where the integration point is located. This means that the corresponding cell number c has to be determined for each integration point. Since a Cartesian grid of hexahedral cells is applied to discretize the embedding domain Ω_e, this cell number can be computed very efficiently. For each integration point, the global coordinates are computed from the mapping function $\mathbf{x}^f = \mathbf{Q}^f(u, v)$ defined by the facet. Based on the global coordinates the corresponding hexahedral can be readily determined, since a structured cell arrangement with fixed spacing is applied. Having found the cell c which contains the present Gaussian point, the local coordinates (ξ, η, ζ) of the cell have to be determined in order to compute the shape function matrix $\mathbf{N}^T(\boldsymbol{\xi})$. It is also possible to compute the inverse mapping, $\boldsymbol{\xi} = \mathbf{Q}^{c-1}(\mathbf{x}^f(u, v))$, very efficiently from global to local cell coordinates, since the mapping (4.19) is linear and can be inverted analytically. One drawback of this way of integration is that the triangular facet is not split up at the intersection with cell boundaries, integrating a function which is not continuous over a surface will loose accuracy even for high order Gaussian integration. Hence a relatively fine mesh is used to reduce integration error to a low level. Note that the numerical effort in integrating over a fine surface mesh with a high order quadrature rule is still relatively small since the two-dimensional integration does not dominate the overall computational effort.

4.4 A CT-derived data specific integration scheme

Rapid and simple grid generation is the major advantage of the FCM and general fictitious domain methods over the standard FEM. The main reason is: in FCM shape functions and model geometry are completely decoupled, so that the Cartesian grid generated in the analysis is not obliged to be aligned to the curved boundaries. As a result, the meshing process is independent of the complexity of the model and is thus straightforward. With trivial effort in mesh generation, the main computational load is shifted to the computation of cell stiffness matrices. As a standard integration approach, Gaussian integration is applied in FCM to compute cell stiffness matrices. For simple structures with homogeneous material properties, only $(p + 1)^3$ integration points are necessary to compute the integration accurately. However, for models with complex geometries and multi-material interfaces the computational cost increases drastically because a dense set of integration points must be allocated inside and outside the physical domain in order to capture both the true physical boundary and the material interfaces. Using this method, considerable amount of computational time has to be spent in order to achieve a high accuracy. One possibility in promoting the performance of the FCM is to accelerate the stiffness matrices integration by precomputation.

To clarify the precomputation procedure, this part of the thesis is structured as follows. Starting from the CT imaging theory given in Section 2.2.1, the three types of geometric representations introduced in [3] will be repeated with emphasis on the last two types in Section 4.4.1. Section 4.4.2 describes the mesh generation method and afterwards, the precomputation scheme is detailed in Section 4.4.3.

4.4.1 Geometric representations

Fast and simply grid generation is one of the main advantages of the FCM. As the starting point of a FCM computation, it is necessary to discuss different ways of representing the geometry of a physical model. Three ways of representation of geometry − implicit representation of geometry, voxel model derived from B-rep representation, and voxel model obtained from a CT scan are accounted for in the FCM [3].

4.4.1.1 Implicit representation of geometry

The implicit representation of the boundary of a geometric model is defined using the zero level set of function $\phi(x) = 0$, similar to *level set* method [78, 79]. Implicit representation gives an exact mathematical description of geometry. The geometric model is responsible for the definition of zero extensions during the integration of cell stiffness matrices. For this purpose, an accurate integration technique is necessary to fully capture the geometry. Different techniques can be utilized to handle this problem, e.g., Gaussian integration with large amount of quadrature points, or the quad-tree based composed integration technique [3]. During the integration, judgment of whether an integration point lies inside or outside is required. This type of representation is, however, mostly applicable in academic problems and is for numerical investigations only. We therefore turn our focus onto accelerating the computations on explicitly represented geometric models.

4.4.1.2 Explicit representation of geometry

One commonly used explicit representation is a voxel model, whose physical boundary is defined by the value of voxels explicitly. Different from models with exact mathematical description of geometry, voxel models provide approximated geometries as well as internal material interfaces. This way of geometric description, on one hand, is geometrically less accurate than the implicit representation; however, it enables a more efficient way of judging whether an integration point is inside or outside the physical domain, since zero extensions are included in the voxel models already. Two types of explicit geometric models have been defined in [3], they are voxel model derived from B-rep representation and voxel model obtained from a CT scan.

4.4.1.2.1 Voxel model derived from B-rep representation

Boundary representation, short for B-rep, is a representation form to describe shapes using a finite number of surfaces. One simple and common B-rep is based on triangulation, through which curved boundaries can be approximated using inter-connected triangles. The STL model, derived from the word STereoLithography [80], is probably the simplest B-rep model which describes the unstructured triangulated surfaces with vertices (coordinate vectors) and surface unit normals. This format of description is employed in FCM to encompass inhomogeneous boundary conditions.

Voxel models can be generated from B-rep models through voxelization. An octree-based voxelization algorithm, which has been implemented in [3, 81] for fast grid generation based

on the hierarchical space-partitioning concept, is adopted to generate voxel models with various resolutions. Figure 4.8 depicts a B-rep model of a telephone handset downloaded from [82] and a voxel model derived from it.

(a) B-rep model (STL mesh with 6,880 triangles) (b) Voxel model with a resolution of $18 \times 18 \times 60$ (c) Voxel model without zero extension

Figure 4.8: Voxelization of a telephone handset

For this type of voxel models, the accuracy of geometric description mainly depends on the voxelization resolution which is adjustable. This feature offers certain amount of flexibility to select a suitable resolution level for problems with various accuracy demands.

4.4.1.2.2 Voxel model derived from a CT scan

In addition to voxelizing B-rep models, a more general approach to obtain voxel models is through CT scanning. The largest difference of the CT-derived voxel models from those obtained from B-rep is that the CT-derived models are originally composed of voxels and contain zero extension already. Hence this type of voxel models is more suitable for the FCM computation, in which voxel data are directly imported and meshed with rectangular grids. It is therefore of great advantage to implement the FCM algorithms in the biomechanical applications, which when solved using the standard finite element method requires large amount of human labor and computational cost in the preprocessing. In comparison with voxel models obtained from B-rep, the downside of CT-derived voxel models is that the model resolution is scanning machine dependent and is for an existing voxel model easier to decrease than to increase. Thus, the flexibility in data adjustment is smaller.

4.4.2 Mesh generation

The QCT-based voxel finite element method (VFEM) [55] is often used to analyze voxel models derived from QCT scans. In VFEM each voxel is converted directly to an eight-node hexahedral element or several voxels are converted to a hexahedral element with an averaged elasticity tensor [83]. Similar to this strategy, in FCM we convert several voxels into one cell which has, rather than an averaged, a piecewise constant elasticity tensor, i.e. the elasticity tensor of each voxel remains unchanged. Since no restriction of number of voxels per cell is defined, the mesh size is relatively flexible: we can have as many/few cells as numerically necessary. Two extremes are that only one cell embeds the whole model or one cell is directly converted from one voxel as in VFEM. Starting from a voxel model derived from either a B-rep model or a CT scan, after fixing the number of voxels in each cell, based on [84] a structured mesh can be automatically generated with trivial effort. A fast meshing algorithm has been developed using 3D index-based reconstruction matrices. See [84] for algorithmic details.

4.4.3 Precomputation of stiffness matrices

For better explaining the idea of this CT-derived data specific precomputation scheme, let us consider a simple example in which the embedding domain Ω_e is meshed with only one cell enclosing $n_x \times n_y \times n_z$ voxels that are extracted directly from a CT scan. The precomputation is carried out in the parametric space where the cell is subdivided into $n_x \times n_y \times n_z$ sub-cells, each can be identified with a voxel from the CT-scan having a constant material property in the Cartesian space. The cell stiffness matrix can be computed by carrying out composed integration, which integrates over $n_x \times n_y \times n_z$ sub-cells. The stiffness matrix of each sub-cell can be precomputed exactly with respect to the elasticity coefficients and unknown sizes of the overlapping voxel. In this way Gaussian integration can be replaced by scalar-matrix multiplication during run-time. The basic formulations for precomputation of general isotropic linear elastic problems are given as follows.

4.4.3.1 Basic formulation

The stiffness matrix formulation

$$\mathbf{K}^c = \sum_{sc=1}^{n_{sc}} \int_t \int_s \int_r (\mathbf{B}^c)^T(\boldsymbol{\xi}(\mathbf{r}))\, \alpha(\mathbf{x}(\boldsymbol{\xi}(\mathbf{r})))\, \mathbf{C}\, \mathbf{B}^c(\boldsymbol{\xi}(\mathbf{r}))\, \det\mathbf{J}^c \det\tilde{\mathbf{J}}_c^{sc}\, drdsdt \tag{4.31}$$

given in Section 4.3 is a general form for the composed integration, which allows for both uniform and non-uniform sub-cell structures. In this fast integration scheme, precomputation with respect to adaptively refined sub-cell structures is not easily feasible due to the fact that the positions of adaptive refinements are model-dependent and are unknown in prior. Hence only uniform sub-cell structures are implemented. Mapping from sub-cells to cell can be avoided and integrations can be carried out directly in the cell parametric space. For simplifying the precomputation, a uniform degree of shape functions is chosen for each cell such that the strain-displacement matrix is the same for all cells ($\mathbf{B}^c = \mathbf{B}$). Equation (4.31) then reads

$$\mathbf{K}^c = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \mathbf{B}^T(\boldsymbol{\xi})\,\alpha(\mathbf{x}(\boldsymbol{\xi}))\,\mathbf{C}\,\mathbf{B}(\boldsymbol{\xi})\,\det\mathbf{J}^c\,d\xi d\eta d\zeta\,. \tag{4.32}$$

The variable α is used to determine whether a point lies in the physical or the extended domain and $\alpha(x(\boldsymbol{\xi}))\mathbf{C}$ denotes the cell material matrix as a function of local coordinates $\boldsymbol{\xi} = (\xi,\,\eta,\,\zeta)^T$. For CT-derived voxel models, the constant property of each voxel allows us to replace $\alpha(\boldsymbol{\xi})\mathbf{C}$ with a piecewise constant matrix function $\bar{\mathbf{C}}^c(\boldsymbol{\xi})$.

Note that one cell embeds $n_x \times n_y \times n_z$ voxels and let us denote the material matrix of each voxel as $\mathbf{C}^c_{ijk}$, where $i = 1\,,...,\,n_x$, $j = 1\,,...,\,n_y$ and $k = 1\,,...,\,n_z$ denote the local voxel indices with respect to ξ, η and ζ directions. Accordingly, $\bar{\mathbf{C}}^c(\boldsymbol{\xi})$ can be written as

$$\bar{\mathbf{C}}^c(\boldsymbol{\xi}) = \mathbf{C}^c_{ijk} \quad \text{when} \quad \begin{cases} T_{\xi,i} \le \xi \le T_{\xi,i+1} \\ T_{\eta,j} \le \eta \le T_{\eta,j+1} \\ T_{\zeta,k} \le \zeta \le T_{\zeta,k+1} \end{cases} \tag{4.33}$$

in which the voxel boundaries are defined as

$$\begin{aligned} T_{\xi,i} &= -1.0 + 2\,\frac{i-1}{n_x}, \quad i = 1,2,...,n_x \\ T_{\eta,j} &= -1.0 + 2\,\frac{j-1}{n_y}, \quad j = 1,2,...,n_y\,. \\ T_{\zeta,k} &= -1.0 + 2\,\frac{k-1}{n_z}, \quad k = 1,2,...,n_z \end{aligned} \tag{4.34}$$

The cell stiffness matrix $\mathbf{K}^c$ then reads

$$\mathbf{K}^c = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} \mathbf{B}^T(\boldsymbol{\xi})\,\bar{\mathbf{C}}^c(\boldsymbol{\xi})\,\mathbf{B}(\boldsymbol{\xi})\,\det\mathbf{J}^c\,d\xi d\eta d\zeta \tag{4.35}$$

where $\det\mathbf{J}^c$ is constant in FCM analysis due to rectangular grids. Inserting (4.33) into (4.35) and by implementing composite integration the stiffness matrix $\mathbf{K}^c$ can be decomposed into summation of $n_x \times n_y \times n_z$ subintegrals

$$\mathbf{K}^c = \sum_{k=1}^{n_z}\sum_{j=1}^{n_y}\sum_{i=1}^{n_x} \mathbf{K}^c_{ijk} \tag{4.36}$$

where

$$\mathbf{K}^c_{ijk} = \int_{T_{\zeta,k}}^{T_{\zeta,k+1}} \int_{T_{\eta,j}}^{T_{\eta,j+1}} \int_{T_{\xi,i}}^{T_{\xi,i+1}} \mathbf{B}^T(\boldsymbol{\xi})\, \mathbf{C}^c_{ijk}\, \mathbf{B}(\boldsymbol{\xi})\, \mathrm{det}\mathbf{J}^c\, d\xi d\eta d\zeta\,. \tag{4.37}$$

For the sake of simplicity, we confine our method to linear isotropic media. In one cell the material matrix of each voxel $\mathbf{C}^c_{ijk}$ can be decomposed as a linear combination of λ^c_{ijk} and μ^c_{ijk} which are the two Lamé constants corresponding to the voxel with index ijk. The relationships between these two constants, elastic modulus and Poisson's ratio read

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}\,, \quad \mu = \frac{E}{2(1+\nu)}\,. \tag{4.38}$$

Decomposition of the voxel material matrix is given as

$$\mathbf{C}^c_{ijk} = \lambda^c_{ijk}\, \mathbf{C}^\lambda + \mu^c_{ijk}\, \mathbf{C}^\mu \tag{4.39}$$

where

$$\mathbf{C}^\lambda = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } \mathbf{C}^\mu = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.40}$$

are constant matrices.

Inserting Equation (4.39) into (4.37) gives

$$\mathbf{K}^c = \sum_{k=1}^{n_z} \sum_{j=1}^{n_y} \sum_{i=1}^{n_x} (\lambda^c_{ijk}\, \mathbf{K}^\lambda_{ijk} + \mu^c_{ijk}\, \mathbf{K}^\mu_{ijk}) \tag{4.41}$$

where

$$\mathbf{K}^\lambda_{ijk} = \int_{T_{\zeta,k}}^{T_{\zeta,k+1}} \int_{T_{\eta,j}}^{T_{\eta,j+1}} \int_{T_{\xi,i}}^{T_{\xi,i+1}} \mathbf{B}^T(\boldsymbol{\xi})\, \mathbf{C}^\lambda\, \mathbf{B}(\boldsymbol{\xi})\, \mathrm{det}\mathbf{J}^c\, d\xi d\eta d\zeta \tag{4.42}$$

and

$$\mathbf{K}^{\mu}_{ijk} = \int_{T_{\zeta,k}}^{T_{\zeta,k+1}} \int_{T_{\eta,j}}^{T_{\eta,j+1}} \int_{T_{\xi,i}}^{T_{\xi,i+1}} \mathbf{B}^T(\boldsymbol{\xi})\, \mathbf{C}^{\mu}\, \mathbf{B}(\boldsymbol{\xi})\, \mathrm{det}\mathbf{J}^c \, d\xi d\eta d\zeta \,. \tag{4.43}$$

denote two matrices corresponding to the voxel with index ijk. These two groups of matrices are independent of the material constants of each voxel, but depend on other parameters, e.g. the voxel dimensions, etc. In order to find out the correlation between the voxel parameters and the two groups of matrices, investigations on each component in Equation (4.42) and (4.43) have been carried out.

Matrices $\mathbf{C}^{\lambda}$ and $\mathbf{C}^{\mu}$ are constant. Computation of the strain-displacement matrix $\mathbf{B}$ requires derivatives of the shape functions $\mathbf{N}$. Computation of derivatives involves the computation of the inverse of Jacobian matrix $\mathbf{J}^c$ which depends on the cell sizes (h_x, h_y, h_z). In terms of (4.34), the integration intervals depend on the number of voxels in each cell n_x, n_y and n_z. Hence, for a given polynomial degree (denoted p) the values of $\mathbf{K}^{\lambda}_{ijk}$ and $\mathbf{K}^{\mu}_{ijk}$ depend on the number of voxels in each direction (n_x, n_y, n_z) and cell sizes (h_x, h_y, h_z). For a CT-derived voxel model, a constant dimension (s_x, s_y, s_z) is assigned to all voxels by default. Therefore, the cell sizes can be computed by multiplying the voxel dimensions (s_x, s_y, s_z) with the number of voxels in each direction ($h_x = s_x\, n_x$, $h_y = s_y\, n_y$, $h_z = s_z\, n_z$). $\mathbf{K}^{\lambda}_{ijk}$ and $\mathbf{K}^{\mu}_{ijk}$ then depend only on six variables n_x, n_y, n_z, s_x, s_y and s_z. As a result, the precomputation of two groups of matrices $\mathbf{K}^{\lambda}$ and $\mathbf{K}^{\mu}$ can be performed once the six variables are determined.

Yet determination of the six parameters involves solving two additional sub-problems. First, the values of n_x, n_y and n_z should be user-definable. Restrictions on the three parameters are required such that the matrices precomputed with respect to the three quantities can be applicable to cells in the entire fictitious domain. Second, since the voxel dimensions s_x, s_y and s_z are CT machine dependent, precomputations with respect to unknown dimensions is not feasible. Therefore a special technique is required to decouple voxel dimensions from precomputed matrices. Solutions to these two problems are presented in Section 4.4.3.2 and 4.4.3.3 separately.

4.4.3.2 Determination of number of voxels per cell

In the current research n_x, n_y and n_z are set to the same integer n_v. With n_v voxels in each direction, one cell contains in total $n_v \times n_v \times n_v$ voxels, denoted by n_{3v}. Each voxel possesses two precomputed matrices $\mathbf{K}^{\lambda}_{ijk}$ and $\mathbf{K}^{\mu}_{ijk}$ of size depending on the polynomial degree of shape functions only. The precomputed matrices can be applied to all cells, when the number of voxels in each direction in the whole fictitious domain can be divided exactly by n_v. Otherwise the original fictitious domain Ω_e has to be extended to a larger fictitious domain denoted by Ω_{e^2} in which the number of voxels in each direction can be divided exactly by n_v. The voxels inside the newly extended domain are set to voids to avoid altering the original physical problem.

4.4.3.3 Precomputation of $\mathbf{K}^{\lambda}$ and $\mathbf{K}^{\mu}$ with respect to s_x, s_y and s_z

Starting from Equation (4.42), the strain-displacement matrix $\mathbf{B}$ is given as

$$\mathbf{B} = \left[\begin{array}{cccccc} \mathbf{B}_1 & \mathbf{B}_2 & \dots & \mathbf{B}_i & \dots & \mathbf{B}_{n_p} \end{array} \right] \tag{4.44}$$

where n_p is the total number of shape functions and the ith strain-displacement submatrix $\mathbf{B}_i$ is given as

$$\mathbf{B}_i = \begin{bmatrix} \frac{\partial N_i(\xi,\eta,\zeta)}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_i(\xi,\eta,\zeta)}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_i(\xi,\eta,\zeta)}{\partial z} \\ \frac{\partial N_i(\xi,\eta,\zeta)}{\partial y} & \frac{\partial N_i(\xi,\eta,\zeta)}{\partial x} & 0 \\ 0 & \frac{\partial N_i(\xi,\eta,\zeta)}{\partial z} & \frac{\partial N_i(\xi,\eta,\zeta)}{\partial y} \\ \frac{\partial N_i(\xi,\eta,\zeta)}{\partial z} & 0 & \frac{\partial N_i(\xi,\eta,\zeta)}{\partial x} \end{bmatrix} \tag{4.45}$$

in which N_i is the ith shape function. Since the shape functions are defined in the local coordinate system, computation of derivatives with respect to the global coordinate system (x, y, z) involves the chain rule:

$$\begin{bmatrix} \frac{\partial N_i(\xi,\eta,\zeta)}{\partial x} \\ \frac{\partial N_i(\xi,\eta,\zeta)}{\partial y} \\ \frac{\partial N_i(\xi,\eta,\zeta)}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} & \frac{\partial \zeta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} & \frac{\partial \zeta}{\partial y} \\ \frac{\partial \xi}{\partial z} & \frac{\partial \eta}{\partial z} & \frac{\partial \zeta}{\partial z} \end{bmatrix} \begin{bmatrix} \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \xi} \\ \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \eta} \\ \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \zeta} \end{bmatrix} = (\mathbf{J}^c)^{-1} \begin{bmatrix} \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \xi} \\ \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \eta} \\ \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \zeta} \end{bmatrix} \tag{4.46}$$

in which the cell Jacobian matrix $\mathbf{J}^c$ is diagonal and depends on the cell sizes.

$$\begin{aligned} \mathbf{J}^c = \begin{bmatrix} \frac{\partial x(\xi,\eta,\zeta)}{\partial \xi} & \frac{\partial y(\xi,\eta,\zeta)}{\partial \xi} & \frac{\partial z(\xi,\eta,\zeta)}{\partial \xi} \\ \frac{\partial x(\xi,\eta,\zeta)}{\partial \eta} & \frac{\partial y(\xi,\eta,\zeta)}{\partial \eta} & \frac{\partial z(\xi,\eta,\zeta)}{\partial \eta} \\ \frac{\partial x(\xi,\eta,\zeta)}{\partial \zeta} & \frac{\partial y(\xi,\eta,\zeta)}{\partial \zeta} & \frac{\partial z(\xi,\eta,\zeta)}{\partial \zeta} \end{bmatrix} &= \frac{1}{2} \begin{bmatrix} h_x & 0 & 0 \\ 0 & h_y & 0 \\ 0 & 0 & h_z \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} s_x\, n_v & 0 & 0 \\ 0 & s_y\, n_v & 0 \\ 0 & 0 & s_z\, n_v \end{bmatrix}. \end{aligned} \tag{4.47}$$

Inverse of the cell Jacobian matrix is given as

$$(\mathbf{J}^c)^{-1} = 2 \begin{bmatrix} \frac{1}{s_x\, n_v} & 0 & 0 \\ 0 & \frac{1}{s_y\, n_v} & 0 \\ 0 & 0 & \frac{1}{s_z\, n_v} \end{bmatrix}. \tag{4.48}$$

Plugging in (4.46) and (4.48) into (4.45) yields

$$\mathbf{B}_i = 2 \begin{bmatrix} \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \xi} \frac{1}{s_x\, n_v} & 0 & 0 \\ 0 & \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \eta} \frac{1}{s_y\, n_v} & 0 \\ 0 & 0 & \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \zeta} \frac{1}{s_z\, n_v} \\ \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \eta} \frac{1}{s_y\, n_v} & \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \xi} \frac{1}{s_x\, n_v} & 0 \\ 0 & \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \zeta} \frac{1}{s_z\, n_v} & \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \eta} \frac{1}{s_y\, n_v} \\ \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \zeta} \frac{1}{s_z\, n_v} & 0 & \frac{\partial N_i(\xi,\eta,\zeta)}{\partial \xi} \frac{1}{s_x\, n_v} \end{bmatrix} . \tag{4.49}$$

As described in Section 2.2.1, each CT image consists of a set of square pixels, namely, all voxel data have equivalent sizes in x, y directions:

$$s_x = s_y \,. \tag{4.50}$$

This property enable us to replace s_y with s_x such that the number of unknown parameters is reduced from three to two − s_x and s_z. s_x and s_z of various voxel models are different from each other − they depend on the CT scanning setting which varies from machine to machine. To overcome the difficulties arising in the precomputation with respect to the two unknown parameters, $\mathbf{K}^\lambda$ and $\mathbf{K}^\mu$ are not directly precomputed but further decomposed.

Inserting (4.50), (4.44) and (4.47) into (4.42) and (4.43), after steps of simplifications $\mathbf{K}^\lambda_{ijk}$ and $\mathbf{K}^\mu_{ijk}$ can be rewritten as

$$\mathbf{K}^\lambda_{ijk} = s_z\, \mathbf{K}^{\lambda_0}_{ijk} + s_x\, \mathbf{K}^{\lambda_1}_{ijk} + \frac{s_x^2}{s_z}\, \mathbf{K}^{\lambda_2}_{ijk} \tag{4.51}$$

$$\mathbf{K}^\mu_{ijk} = s_z\, \mathbf{K}^{\mu_0}_{ijk} + s_x\, \mathbf{K}^{\mu_1}_{ijk} + \frac{s_x^2}{s_z}\, \mathbf{K}^{\mu_2}_{ijk} \tag{4.52}$$

Introducing a new parameter $r = s_x/s_z$, (4.51) and (4.52) then read

$$\mathbf{K}^\lambda_{ijk} = s_z\, (\mathbf{K}^{\lambda_0}_{ijk} + r\, \mathbf{K}^{\lambda_1}_{ijk} + r^2\, \mathbf{K}^{\lambda_2}_{ijk}) \tag{4.53}$$

$$\mathbf{K}^\mu_{ijk} = s_z\, (\mathbf{K}^{\mu_0}_{ijk} + r\, \mathbf{K}^{\mu_1}_{ijk} + r^2\, \mathbf{K}^{\mu_2}_{ijk}) \tag{4.54}$$

where $\mathbf{K}^{\lambda_0}_{ijk}$, $\mathbf{K}^{\lambda_1}_{ijk}$, $\mathbf{K}^{\lambda_2}_{ijk}$, $\mathbf{K}^{\mu_0}_{ijk}$, $\mathbf{K}^{\mu_1}_{ijk}$ and $\mathbf{K}^{\mu_2}_{ijk}$ denote six groups of constant matrices depending only on p and n_v. Each matrix is symmetric, densely populated and has the same size as $\mathbf{K}^{\mathbf{c}}$. With respect to different p and n_v, these matrices can be precomputed and stored.

Several existing methods can be applied to precompute the matrices accurately, e.g., Gaussian quadratures, adaptive trapezoid integration, or adaptive Simpson integration. In our precomputation, hierarchic shape functions that span the trunk space are employed to minimize the matrix size. Some information of the precomputed matrices $\mathbf{K}^{\lambda_0}$ ($n_v = 10$) is given in Table 4.1.

Polynomial degree	1	2	3	4	5	6
Number of rows/columns	24	60	96	150	222	315
Data size (Mbyte)	2.29	13.96	35.52	86.40	188.85	379.71

Table 4.1: Precomputed matrix size and data size for $\mathbf{K}^{\lambda_0}$ with $n_v = 10$

It is obvious that a large reduction of necessary memory-space could be obtained by using symmetry of the coordinate axes. It follows immediately, that all matrices K_{ijk} can be obtained by row and column permutation of i, j and k. Therefore only single matrices would have to be stored. This would reduce the necessary memory space by $3! = 6$. Further reduction would be possible (by another factor of 8), if one takes into account that $K_{n_v-i,j,k}$ can be immediately obtained from K_{ijk}, and the same for other indices. Summarizing the overall memory space could be reduced by a factor of approximately 50. This possible reduction is not implemented for two reasons. First, it would require a lot additional effort to find out the permutation matrices with which the matrices in (4.53) and (4.54) can be reconstructed. Second, in Section 4.4.6 the computation of stiffness matrices is accelerated by using the BLAS routine. For this implementation, the two groups of matrices $\mathbf{K}^{\lambda}_{ijk}$ and $\mathbf{K}^{\mu}_{ijk}$ in (4.53) and (4.54) are required to be completely stored in the memory. Furthermore, in Section 5.3.5.2 these two groups of matrices are also required to be stored in a complete form to ensure an efficient numerical simulation during the entire computational steering process. As one of the major goals of the research was the speed of computation, reconstructing new matrices from a condensed form involves additional memory references and data movement and was thus not accepted for current applications. Moreover, as these matrices occupy only a small fraction of the computer memory-space (less than 5% for $n_v = 10$ and $p = 6$ on a modern computer with 16Gb physical memory), its negative impact can be neglected in this research.

4.4.4 Stiffness matrices computing procedure with precomputed matrices

With precomputed matrices at hand, the procedure of stiffness matrices computation is illustrated in Figure 4.9. The polynomial degree p and the number of voxels per cell n_v are user defined input parameters. After reading in the CT data the program generates rectangular grids and computes r and s_z. Subsequently it reads in six groups of precomputed matrices and computes $\mathbf{K}^{\lambda}$ and $\mathbf{K}^{\mu}$. In the next step, the cell stiffness matrices are computed by performing scalar-matrix multiplications.

4.4.5 Computational efficiency estimation

To estimate the speedup factor by using the precomputed matrices, the number of floating point operations in computing the stiffness matrix of one cell using the FCM without precomputation is firstly estimated. A cell is assumed to contain $n_v \times n_v \times n_v$ sub-cells, in each sub-cell $(p+1)^3$ Gaussian integration points are adopted to integrate exactly. Thanks to the symmetry, only entries in the upper or lower triangular part of stiffness matrix have to be

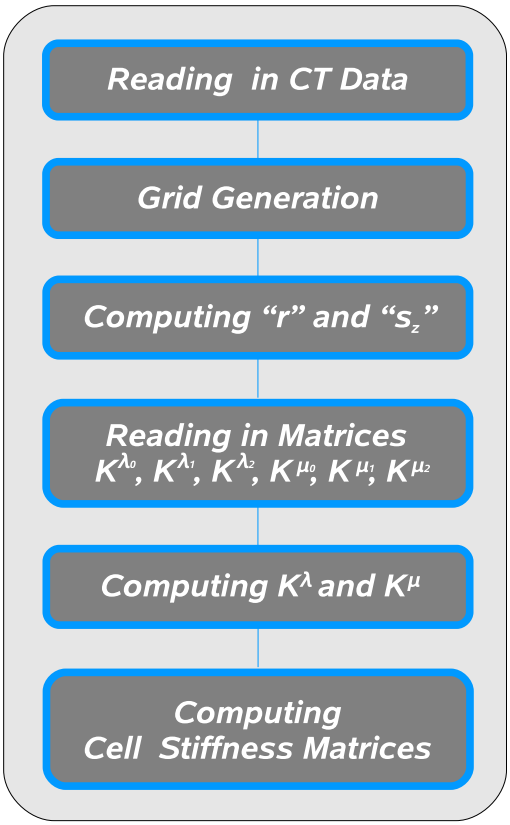


Figure 4.9: Stiffness matrices computing procedure

computed. Denoting the number of rows of matrix $\mathbf{K}^c$ as n_r, the total number of entries in the lower triangular part of the matrix n_t, we have $n_t = \frac{1}{2}\, n_r\, (n_r + 1)$. The estimated number of floating point operations is

$$T_1 = 6\, n_r\, (6 + n_r)\, (p+1)^3\, n_{3v} \tag{4.55}$$

and for FCM with precomputed matrices the number of operations is

$$T_2 = 2\, n_t\, n_{3v}\,. \tag{4.56}$$

The speedup factor is $T_1/T_2 \approx 6\,(p+1)^3$, which for $p = 6$ equals 2,058.

4.4.6 Acceleration of scalar-matrix computation using the BLAS routine

To effectively utilize a high-performance computer, it is significant to avoid unnecessary memory references. The movement of data between memory and registers can be as costly as floating point operations [85], and accordingly the efficiency of many linear algebra computations can be achieved not by developing new computing algorithms, but rather by compiling algorithms to minimize data movement and maximize the reuse of data in cache.

4.4.6.1 BLAS

One possible approach of improving the efficiency is to use the Basic Linear Algebra Subprograms (BLAS), which are routines that provide standard building blocks for performing basic vector and matrix operations [85]. Highly optimized implementations of the BLAS interface have been developed by hardware vendors such as Intel and AMD, as well as by other authors, e.g. Goto BLAS and ATLAS [86]. The BLAS package includes FORTRAN routines of three levels, the level 1 BLAS performs vector-vector operations; the level 2 BLAS performs matrix-vector operations; the level 3 BLAS performs matrix-matrix operations. The efficiency of the BLAS routines are indicated by the ratio of floating-point operations to data movement, which if high enough, shows an efficient reuse of data in cache. The level 3 BLAS that targets at matrix-matrix operations can achieve very good performance on high-performing computers with memory hierarchy. The reason is that the matrices are partitioned into blocks on which the matrix-matrix operations are performed. This way the block data which is held in cache can be fully reused and in addition the operations on blocks enable the parallelization on multi-core computers. With these considerations, on top of the precomputation described in Section 4.4.3 the stiffness matrix computation can be speeded up the second time if the scalar-matrix multiplications can be transformed to matrix-matrix multiplications.

4.4.6.2 Matrix transformation

To facilitate the description, the total number of cells is denoted as n_c. For convenience matrices $\mathbf{K}^{\lambda}_{ijk}$ and $\mathbf{K}^{\mu}_{ijk}$ are replaced with $\mathbf{K}^{\lambda}_{r}$ and $\mathbf{K}^{\mu}_{r}$ $(r = 1, ..., n_v^3)$ by reorganizing the indices from 3D to 1D. Accordingly, Equation (4.41) becomes

$$\mathbf{K}^c = \sum_{r=1}^{n_{3v}} \lambda_r^c \, \mathbf{K}_r^{\lambda} + \sum_{r=1}^{n_{3v}} \mu_r^c \, \mathbf{K}_r^{\mu} \tag{4.57}$$

In the next step $\mathbf{K}^{\lambda}_{r}$, $\mathbf{K}^{\mu}_{r}$ and $\mathbf{K}^{c}$ are vectorized from matrices to single column vectors in a column-majored order. For reducing vector size, only entries in the lower triangular part of the matrix are considered. The new vectors are denoted as $\bar{\mathbf{K}}^{\lambda}_{r}$, $\bar{\mathbf{K}}^{\mu}_{r}$ and $\bar{\mathbf{K}}^{c}$, respectively. The size of vector $\bar{\mathbf{K}}^{\lambda}_{r}$ and $\bar{\mathbf{K}}^{\mu}_{r}$ is denoted as n_t, which equals to $\frac{1}{2}\, n_r \,(n_r + 1)$ where n_r is the number of rows of $\mathbf{K}^c$.

Thereafter, Equation (4.57) is rewritten as

$$\bar{\mathbf{K}}^c = \begin{bmatrix} \bar{\mathbf{K}}^c_1 \\ \bar{\mathbf{K}}^c_2 \\ . \\ . \\ . \\ \bar{\mathbf{K}}^c_{n_t} \end{bmatrix} = \lambda^c_1 \begin{bmatrix} \bar{\mathbf{K}}^\lambda_{11} \\ \bar{\mathbf{K}}^\lambda_{12} \\ . \\ . \\ . \\ \bar{\mathbf{K}}^\lambda_{1n_t} \end{bmatrix} + \lambda^c_2 \begin{bmatrix} \bar{\mathbf{K}}^\lambda_{21} \\ \bar{\mathbf{K}}^\lambda_{22} \\ . \\ . \\ . \\ \bar{\mathbf{K}}^\lambda_{2n_t} \end{bmatrix} + ... + \lambda^c_{n_{3v}} \begin{bmatrix} \bar{\mathbf{K}}^\lambda_{n_{3v}1} \\ \bar{\mathbf{K}}^\lambda_{n_{3v}2} \\ . \\ . \\ . \\ \bar{\mathbf{K}}^\lambda_{n_{3v}n_t} \end{bmatrix} + \mu^c_1 \begin{bmatrix} \bar{\mathbf{K}}^\mu_{11} \\ \bar{\mathbf{K}}^\mu_{12} \\ . \\ . \\ . \\ \bar{\mathbf{K}}^\mu_{1n_t} \end{bmatrix} + \mu^c_2 \begin{bmatrix} \bar{\mathbf{K}}^\mu_{21} \\ \bar{\mathbf{K}}^\mu_{22} \\ . \\ . \\ . \\ \bar{\mathbf{K}}^\mu_{2n_t} \end{bmatrix} + ... + \mu^c_{n_{3v}} \begin{bmatrix} \bar{\mathbf{K}}^\mu_{n_{3v}1} \\ \bar{\mathbf{K}}^\mu_{n_{3v}2} \\ . \\ . \\ . \\ \bar{\mathbf{K}}^\mu_{n_{3v}nt} \end{bmatrix} . \tag{4.58}$$

Transforming Equation (4.58) into matrix-vector multiplication form gives

$$\bar{\mathbf{K}}^c = \begin{bmatrix} \bar{\mathbf{K}}^\lambda_{11} & \bar{\mathbf{K}}^\lambda_{21} & . & . & . & \bar{\mathbf{K}}^\lambda_{n_{3v}1} \\ \bar{\mathbf{K}}^\lambda_{12} & \bar{\mathbf{K}}^\lambda_{22} & . & . & . & \bar{\mathbf{K}}^\lambda_{n_{3v}2} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \bar{\mathbf{K}}^\lambda_{1n_t} & \bar{\mathbf{K}}^\lambda_{2n_t} & . & . & . & \bar{\mathbf{K}}^\lambda_{n_{3v}n_t} \end{bmatrix} \begin{bmatrix} \lambda^c_1 \\ \lambda^c_2 \\ . \\ . \\ . \\ \lambda^c_{n_{3v}} \end{bmatrix} + \begin{bmatrix} \bar{\mathbf{K}}^\mu_{11} & \bar{\mathbf{K}}^\mu_{21} & . & . & . & \bar{\mathbf{K}}^\mu_{n_{3v}1} \\ \bar{\mathbf{K}}^\mu_{12} & \bar{\mathbf{K}}^\mu_{22} & . & . & . & \bar{\mathbf{K}}^\mu_{n_{3v}2} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \bar{\mathbf{K}}^\mu_{1n_t} & \bar{\mathbf{K}}^\mu_{2n_t} & . & . & . & \bar{\mathbf{K}}^\mu_{n_{3v}n_t} \end{bmatrix} \begin{bmatrix} \mu^c_1 \\ \mu^c_2 \\ . \\ . \\ . \\ \mu^c_{n_{3v}} \end{bmatrix} . \tag{4.59}$$

By assembling $\bar{\mathbf{K}}^{\mathbf{c}}$ ($c = 1\,, ...,\, n_c$) of each cell in terms of cell numbers into a matrix denoted as $\mathbf{K}^{\mathbf{G}}$, the matrix-vector multiplications are transformed into two matrix-matrix multiplications which can be efficiently computed with the help of the BLAS subroutine "**dgemm**", see Equation (4.60).

$$
\begin{aligned}
\mathbf{K}^G &= \left[\begin{array}{cccccc} \bar{\mathbf{K}}^1 & \bar{\mathbf{K}}^2 & . & . & . & \bar{\mathbf{K}}^{n_c} \end{array}\right] = \left[\begin{array}{ccccc} \bar{\mathbf{K}}_1^1 & \bar{\mathbf{K}}_1^2 & . & . & . & \bar{\mathbf{K}}_1^{n_c} \\ \bar{\mathbf{K}}_2^1 & \bar{\mathbf{K}}_2^2 & . & . & . & \bar{\mathbf{K}}_2^{n_c} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \bar{\mathbf{K}}_{n_t}^1 & \bar{\mathbf{K}}_{n_t}^2 & . & . & . & \bar{\mathbf{K}}_{n_t}^{n_c} \end{array}\right] \\
&= \left[\begin{array}{ccccc} \bar{\mathbf{K}}_{11}^{\lambda} & \bar{\mathbf{K}}_{21}^{\lambda} & . & . & . & \bar{\mathbf{K}}_{n_{3v}1}^{\lambda} \\ \bar{\mathbf{K}}_{12}^{\lambda} & \bar{\mathbf{K}}_{22}^{\lambda} & . & . & . & \bar{\mathbf{K}}_{n_{3v}2}^{\lambda} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \bar{\mathbf{K}}_{1n_t}^{\lambda} & \bar{\mathbf{K}}_{2n_t}^{\lambda} & . & . & . & \bar{\mathbf{K}}_{n_{3v}n_t}^{\lambda} \end{array}\right] \left[\begin{array}{ccccc} \lambda_1^1 & \lambda_1^2 & . & . & . & \lambda_1^{n_c} \\ \lambda_2^1 & \lambda_2^2 & . & . & . & \lambda_2^{n_c} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \lambda_{n_{3v}}^1 & \lambda_{n_{3v}}^2 & . & . & . & \lambda_{n_{3v}}^{n_c} \end{array}\right] \\
&+ \left[\begin{array}{ccccc} \bar{\mathbf{K}}_{11}^{\mu} & \bar{\mathbf{K}}_{21}^{\mu} & . & . & . & \bar{\mathbf{K}}_{n_{3v}1}^{\mu} \\ \bar{\mathbf{K}}_{12}^{\mu} & \bar{\mathbf{K}}_{22}^{\mu} & . & . & . & \bar{\mathbf{K}}_{n_{3v}2}^{\mu} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \bar{\mathbf{K}}_{1n_t}^{\mu} & \bar{\mathbf{K}}_{2n_t}^{\mu} & . & . & . & \bar{\mathbf{K}}_{n_{3v}n_t}^{\mu} \end{array}\right] \left[\begin{array}{ccccc} \mu_1^1 & \mu_1^2 & . & . & . & \mu_1^{n_c} \\ \mu_2^1 & \mu_2^2 & . & . & . & \mu_2^{n_c} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \mu_{n_{3v}}^1 & \mu_{n_{3v}}^2 & . & . & . & \mu_{n_{3v}}^{n_c} \end{array}\right]
\end{aligned}
\tag{4.60}
$$

With the above approach, the stiffness matrices of all cells are computed altogether at the beginning right after $\mathbf{K}_{ijk}^{\lambda}$ and $\mathbf{K}_{ijk}^{\mu}$ are formed in terms of Equation (4.51) and (4.52). Afterwards, the stiffness matrix of each cell can be retrieved from $\mathbf{K}^G$.

The speedup factor of using the BLAS routine varies from problem to problem, it does not only depend on the matrix size but also on the number of cores, cache sizes and so on. To provide an approximate quantification of this factor, a series of test computations have been carried out on an Intel Xeon W5590, 3.33GHz work station, 8 cores. The setting of $10 \times 10 \times 10$ voxels per cell ($n_{3v} = 1000$) is used in order to minimize the total number of cells. With this setting, the speedup factor using the BLAS routine of the Intel-MKL library for $p = 1, ..., 6$ and $n_c = 1, ..., 2000$ was evaluated. Only one core is activated during the entire evaluation. Graphs plotted for all polynomial degrees have a similar distribution which starts from approximately 3 and increases up to 6. For the reason of clarity, only the graphs corresponding to $p = 4$ and $p = 5$ are plotted in Figure 4.10.

From reading the graphs, one can see that the speedup factor is, rather than a fixed value, a range of from 3 to 6. Multiplying this range with $6\,(p+1)^3$ – the speedup factor obtained in Section 4.4.5, the final speedup factor varies from $18\,(p+1)^3$ to $36\,(p+1)^3$. For $p = 6$ the maximum speedup tends to 12,348. It is important to mention that the code for stiffness matrix computation can be easily and efficiently parallelized on a multi-core machine using OpenMP. The efficiency of parallelization is demonstrated in the third example in Section 4.5.

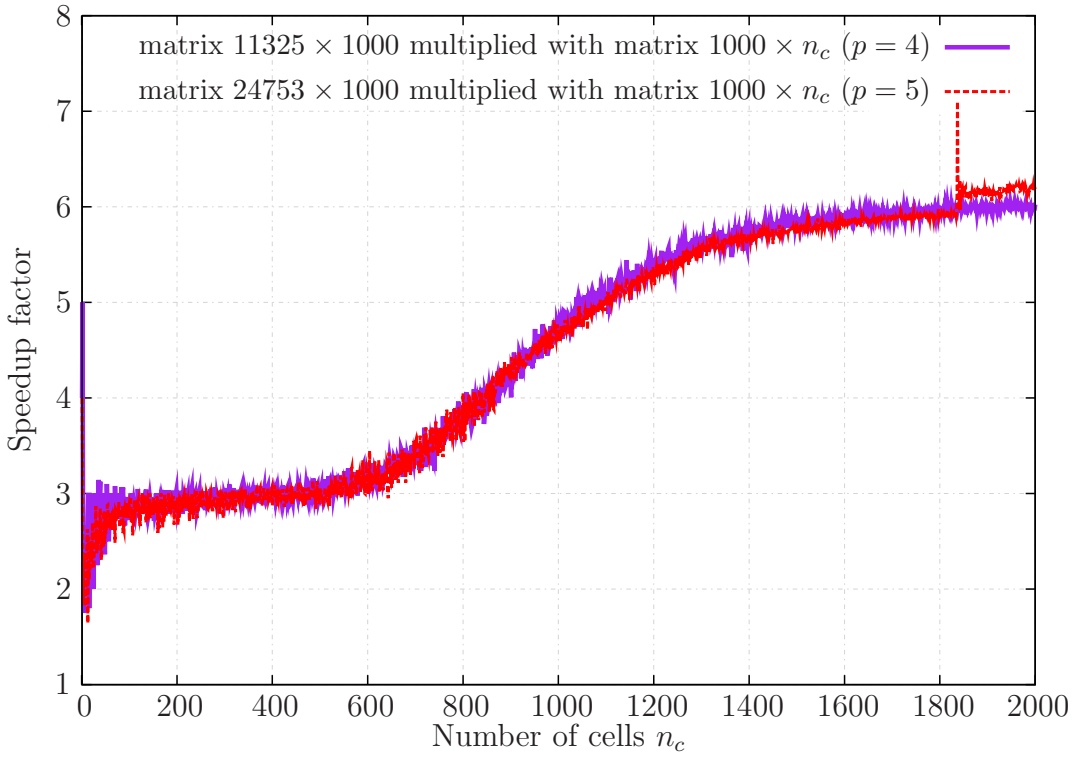


Figure 4.10: Speedup factor graphs evaluated for $p = 4$ and $p = 5$

4.5 Numerical examples

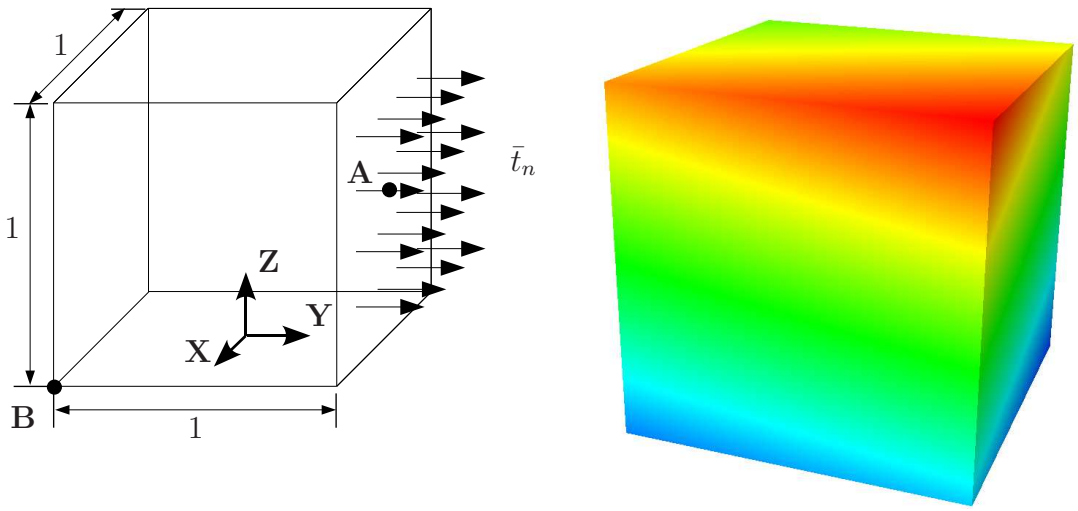
In this section, four numerical examples are given for verifying the FCM with fast integration scheme and, moreover, to demonstrate its numerical efficiency. The computations are carried out using a p-FEM code named "AdhoC" [87] which has been modified to adapt to the fast integration scheme. The number of voxels per cell is set to either $5 \times 5 \times 5$ or $10 \times 10 \times 10$ for all problems. The direct solver "Pardiso" [88, 89] is adopted as the solver for FCM due to its numerical stability and good parallelizability. In the post-processing $15 \times 15 \times 15$ resulting points are set within one cell. Results on these points are visualized with the open-source visualization tool "Paraview" [90].

4.5.1 Inhomogeneous unit cube

In this example the accuracy of the FCM computation is verified by a unit cube with inhomogeneous material property. The cube model and its problem setup are given in Figure 4.11. The size of the cube is $1 \times 1 \times 1mm$. The material model is assumed to be linear elastic, isotropic and inhomogeneous with a constant Poisson's ratio $\nu = 0.3$ and a varying Young's modulus described by

$$E = (x+10)^2 (y+10) (z+1) N/mm^2 \tag{4.61}$$

which is visualized in Figure 4.11(b). Symmetry boundary conditions are applied at $x = -0.5, y = -0.5, z = 0$ mm, while a traction load with the value of $1.0N/mm^2$ is applied in the normal direction on the face of $y = 0.5$, see Figure 4.11(a).



(a) The dimensions of the cube model and the problem setup

(b) Plot of Young's modulus variation within the cube

Figure 4.11: A unit cube model

A numerical analysis of this model was performed using the standard p-version finite element method in [73], in which a mesh of $2 \times 2 \times 2$ is used while in each element the material matrix is approximated by high-order polynomials. A uniform p-extension was conducted with

$p = 1, ..., 8$. $p + 1$ Gaussian integration points are used in each element to attain highly accurate results which are taken as the reference solution for the FCM computation.

In the FCM computation, a voxel model is obtained by voxelizing the solid model into $20 \times 20 \times 20$ voxels, each has a constant but different value of Young's modulus which is computed by evaluating Equation (4.61) at the center of each voxel, see Figure 4.12. With $10 \times 10 \times 10$ voxels per cell, the voxel model is embedded into $2 \times 2 \times 2$ cells shown in the same figure.

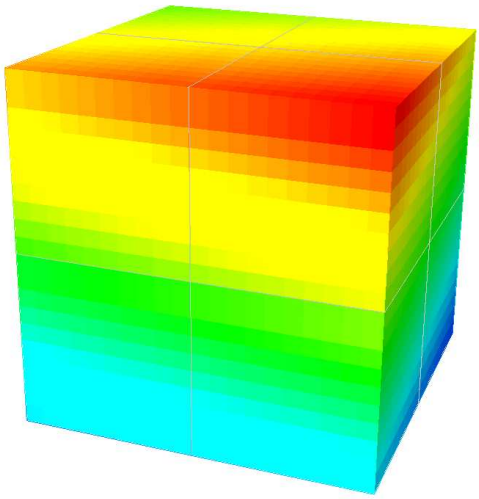


Figure 4.12: Voxel model $20 \times 20 \times 20$ and its discretization

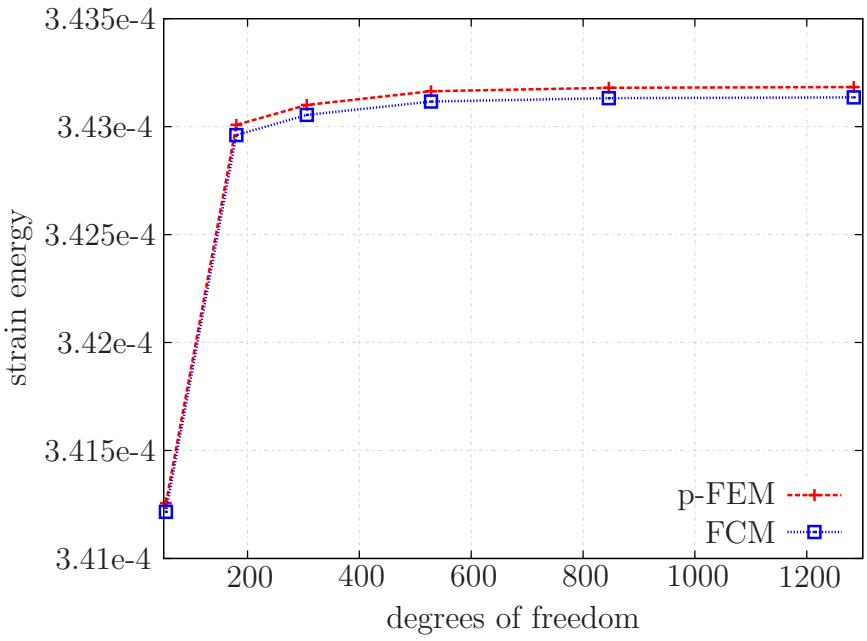


Figure 4.13: Strain energy convergence graph

The FCM analysis is conducted by a uniform p-extension with $p = 1, ..., 6$ using the fast integration scheme. The strain energy of the p-FEM and the FCM solution are plotted in Figure 4.13. The relative difference of the two strain energy at $p = 6$ is about 0.014%.

The displacement distribution in z-direction and the strain distribution in y-direction are plotted in Figure 4.14.

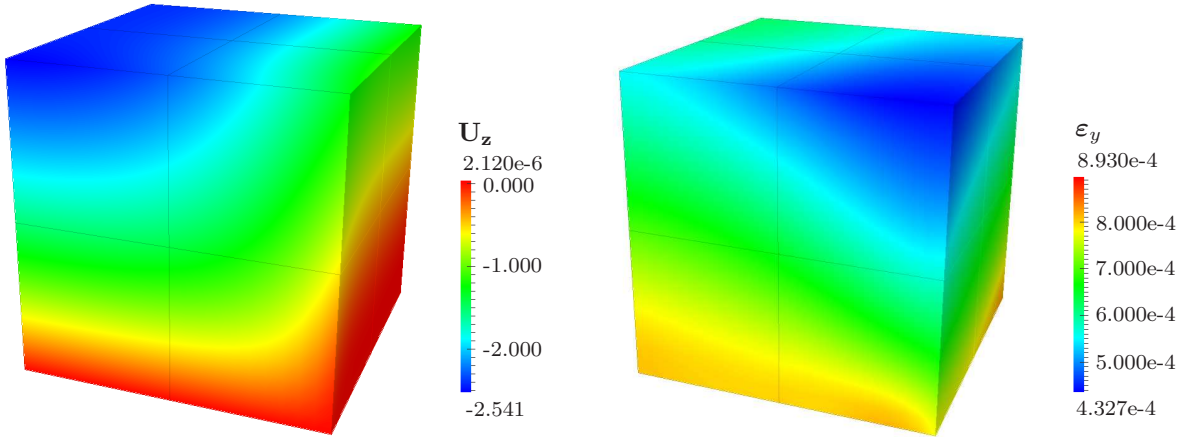


Figure 4.14: Displacement distribution in z-direction (U_z) and strain distribution in y-direction (ε_y) for $p = 4$ on the mesh of $2 \times 2 \times 2$

In addition to strain energy, the displacement results in z-direction evaluated at point A $(0.0, 0.5, 0.5)$ and strain results in y-direction evaluated at point B $(0.5, -0.5, 0.0)$ are compared by using the two methods. The two evaluation points are indicated in Figure 4.11(a). The displacement and strain plots are shown in Figure 4.15 and 4.16.

The FCM results coincide very well with the p-FEM results. For $p = 6$, the relative difference in displacement at point A is approximately 0.14%; while the relative difference in strain at point B is about 0.16%.

It is noteworthy that the voxel model for FCM computation is not the "exact" model which is used in the p-FEM computation. Geometrically, the voxel model exactly represents the original model; however, the deviation in material distribution caused by the voxelization process generates a modeling error. One can read from the results that the impact of material modeling error on the accuracy of the FCM computation is limited.

4.5.2 Thin-walled plate with a circular hole

The thin-walled plate with a circular hole example is a widely accepted benchmark in computational mechanics. In Figure 4.17 only one eighth of a plate with a hole in the center is presented. The size of the plate is $10 \times 10 \times 1mm$ and the radius of the hole is 1mm. The plate material is assumed to be linear homogeneous isotropic with Young's modulus $E = 206900N/mm^2$ and Poisson's ratio $\nu = 0.29$. Symmetry boundary conditions are applied at $x = y = z = 0$ mm and a traction load of 100 N/mm^2 is imposed as indicated in Figure 4.17. As the reference solution obtained by an "overkill" finite element approximation the

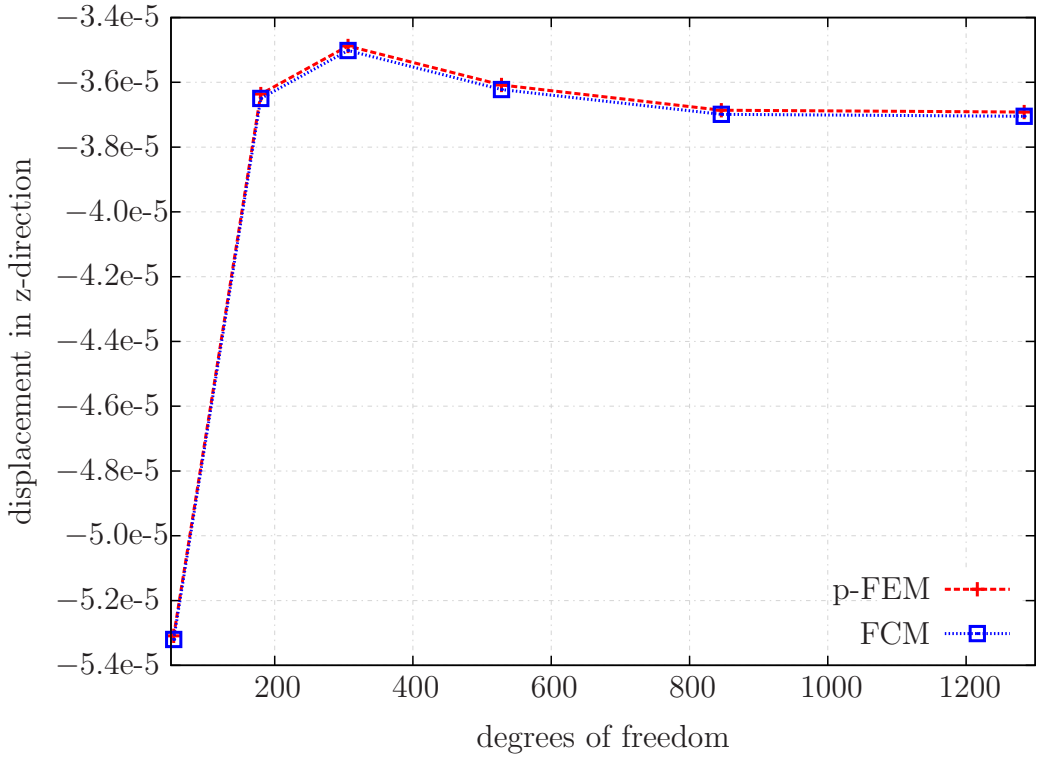


Figure 4.15: Displacement in z-direction computed at point A by p-FEM and FCM

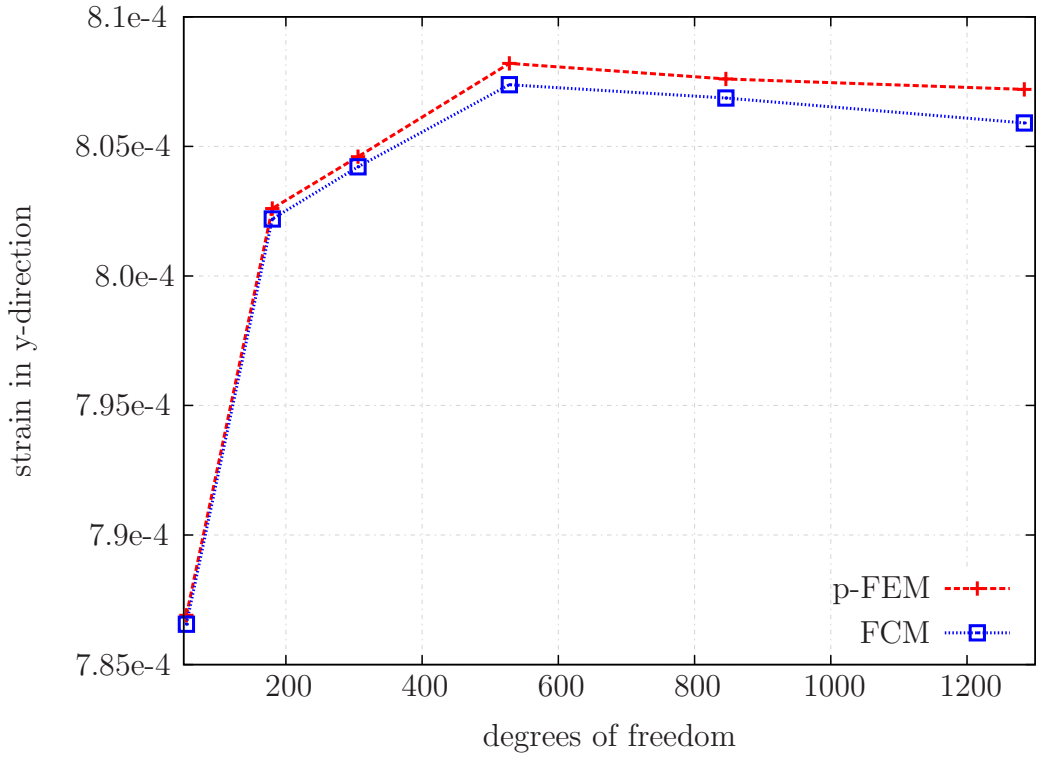


Figure 4.16: Strain in y-direction computed at point B by p-FEM and FCM

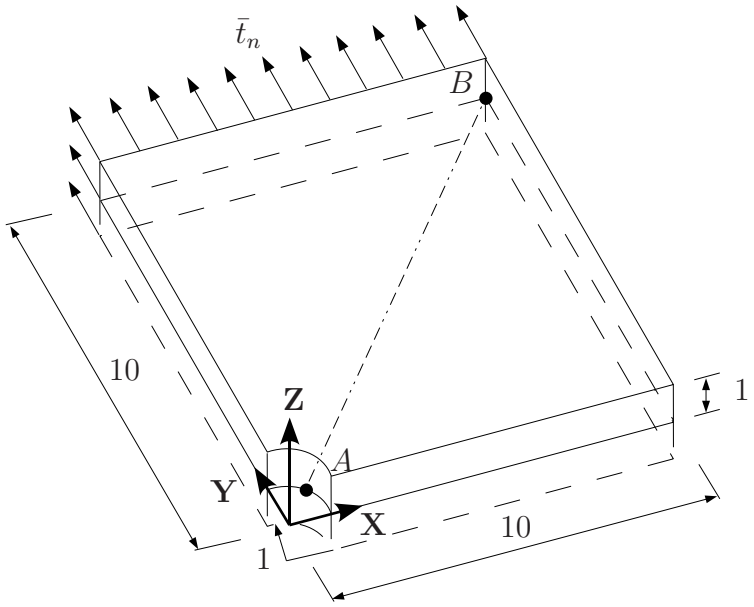


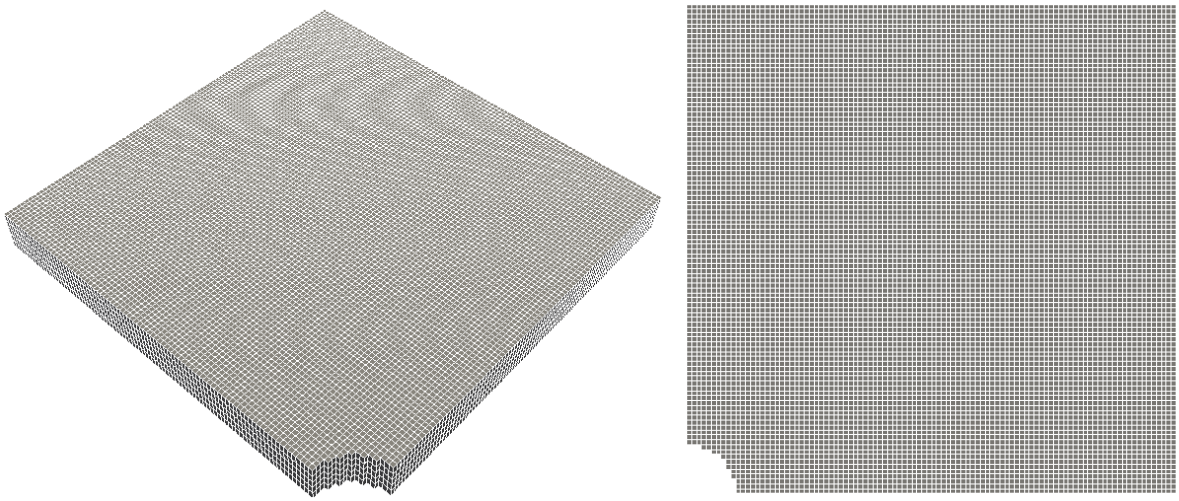
Figure 4.17: Thin-walled plate with circular hole

strain energy of the plate amounts to 2.475213962 [69].

The geometry of this implicitly represented model can be exactly described with mathematical functions, especially the circular boundary. Using adaptive sub-cell structures, it is possible to precisely locate a dense group of Gaussian integration points near the physical boundary to capture the exact geometry. However, performing fast integration by accounting whether a Gaussian point locates inside or outside the physical domain is no longer feasible, since no Gaussian points are used. The geometric representation is transformed from implicit to explicit and thereafter one voxel is regarded as a homogeneous isotropic solid even when it intersects with the real physical boundary. In this case, modeling errors are generated and a quantitative error analysis is necessary.

A C++ program has been written to voxelize the model with resolutions as user-defined input parameters. The zero extension is defined by judging whether the center of a voxel lies inside or outside the geometric model. Figure 4.18 depicts one voxel model with a resolution of $100 \times 100 \times 5$. For vertical or horizontal surfaces the voxelized model can fully represent the original model without introducing any geometric deviation. However, for oblique or curved surface the voxelization process induces modeling errors caused by deviation of the jagged boundary from the oblique or curved surface.

A schematic representation of a voxel model generated by voxelization of a quarter circle is shown in Figure 4.19(a) while its geometric deviations are highlighted in Figure 4.19(b). In Figure 4.19(b) the regions marked in dark grey stand for the geometric deviations overestimated by voxel representation, while the regions in light grey represent the geometric deviations underestimated. These two types of differences account for the modeling errors of the voxel model. To quantify the errors, we evaluate in this example the relative error in



(a) Isometric view (b) Top view

Figure 4.18: Voxel model with a resolutions of $100 \times 100 \times 5$

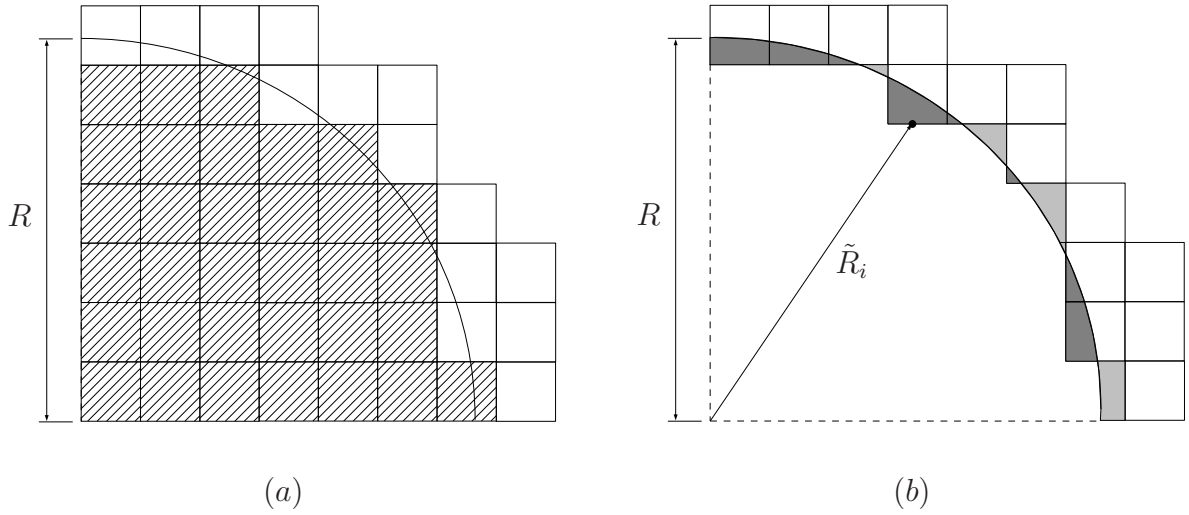


Figure 4.19: Schematic representation of a voxel model and its geometric deviations at a curved boundary

radius, which is computed using Equation (4.62). $\tilde{R}_i$ is evaluated at $n_s = 500 * n_{edges}$ (n_{edges} denotes the number of edges involved) sampling points equally distributed on the edges of the jagged boundary which approximates the circular arc, see Figure 4.19(b). L_2 norm is used in the computation in order to prevent cancellation between the positive and negative parts.

$$(\bar{e}_R)_{E(\Omega)} = \left| \frac{R - \tilde{R}_i}{R} \right|_{L_2} 100[\%] \tag{4.62}$$

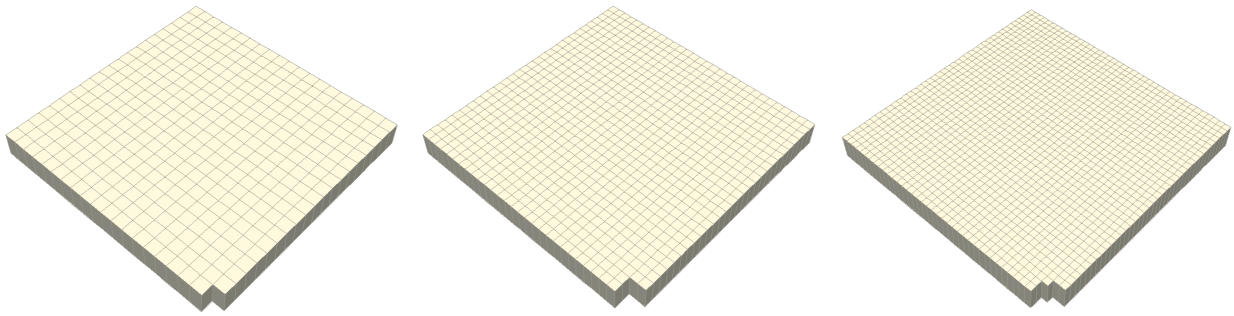
For convergence studies the voxel model is refined in the xy-plane, in addition, three voxel models with resolutions of $200 \times 200 \times 10$, $300 \times 300 \times 10$ and $400 \times 400 \times 10$ are generated.

Relative errors of the four models have been estimated by applying a reference solution and are listed in Table 4.2.

Number	**Resolution**	**Error[%]**
1	$100 \times 100 \times 5$	2.783691
2	$200 \times 200 \times 10$	1.433703
3	$300 \times 300 \times 10$	0.900102
4	$400 \times 400 \times 10$	0.679198

Table 4.2: Resolutions and corresponding modeling errors

From voxel models cell grids are generated. Three sample meshes are shown in Figure 4.20.



(a) $20 \times 20 \times 1$ (b) $30 \times 30 \times 1$ (c) $40 \times 40 \times 1$

Figure 4.20: Hexahedral meshes of voxel models with different cell resolutions

For the voxel models $100 \times 100 \times 5$, each cell represents $5 \times 5 \times 5$ voxels; while $10 \times 10 \times 10$ voxels per cell is applied to the voxel models $200 \times 200 \times 10$, $300 \times 300 \times 10$ and $400 \times 400 \times 10$.

As the model is refined the voxel size decreases as well as the hexahedral grid size, which results in more cells lying completely outside the circular hole. These cells are discarded to save computational cost, see Figure 4.20(a), 4.20(b) and 4.20(c). The Dirichlet and Neumann Boundary conditions are applied directly onto the cell boundaries. With the fast integration scheme a series of computations have been carried out. In this example only p-refinement is considered, the polynomial degree for all cases increases uniformly from one to six.

The strain energy against the degrees of freedom is plotted in Figure 4.21.

In terms of Equation (4.63) [3]

$$(e_r)_{E(\Omega)} = \sqrt{\frac{|\mathcal{B}(\mathbf{u}_{EX}, \mathbf{u}_{EX}) - \mathcal{B}_e(\mathbf{u}_{FC}, \mathbf{u}_{FC})|}{\mathcal{B}(\mathbf{u}_{EX}, \mathbf{u}_{EX})}} 100[\%] \tag{4.63}$$

the relative error in energy norm is plotted versus the degrees of freedom in Figure 4.22.

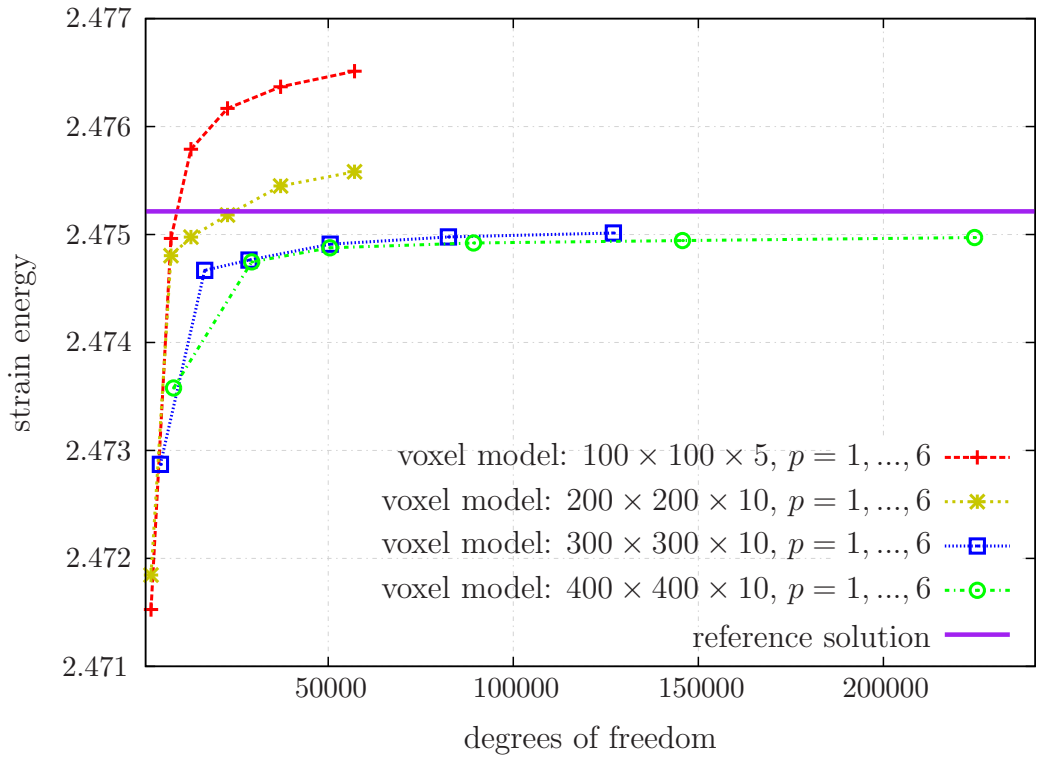


Figure 4.21: Strain energy convergence graph

As each voxel model represents a (slightly) different geometry, the strain energy of the four models converges to different values as the polynomial degree increases (see Figure 4.21). The converged values of the two coarser voxel models are higher than the reference solution. For models with resolution $300 \times 300 \times 10$ and $400 \times 400 \times 10$ the relative errors in strain energy reduce to below 1% as the polynomial degree increases. Nevertheless the exponential rate of convergence as in [3], where the exact geometry has been considered by means of an implicit representation, can not be observed. Since the geometry of the circular arc is different from that with the voxelized arc, accordingly the "exact solution" on a voxelized model is different from the one in [3]. As the voxel model is refined, the modeling error reduces and subsequently the FCM solution approximates more closely the "exact solution". It is noteworthy that the error in energy norm conforms to the modeling error tabulated in Table 4.2.

Figure 4.23 depicts the von Mises stress distribution computed on the voxel model $300 \times 300 \times 10$ with a mesh of $30 \times 30 \times 1$ and $p = 4$. The stresses were evaluated on $15 \times 15 \times 15$ postprocessing points per cell. The visualization tool "Paraview" visualized the result directly without specific postprocessing approach employed.

In this example, voxels that approximate the circular hole form a jagged boundary which yields stress singularities along the circular arc. To further investigate the influence of singularity, von Mises stresses along the cutline A-B were surveyed. The reference solution obtained by an "overkilled" FE analysis is plotted in Figure 4.24. The relative errors of the FCM solutions have been computed and are plotted in logarithmic scale in Figure 4.25.

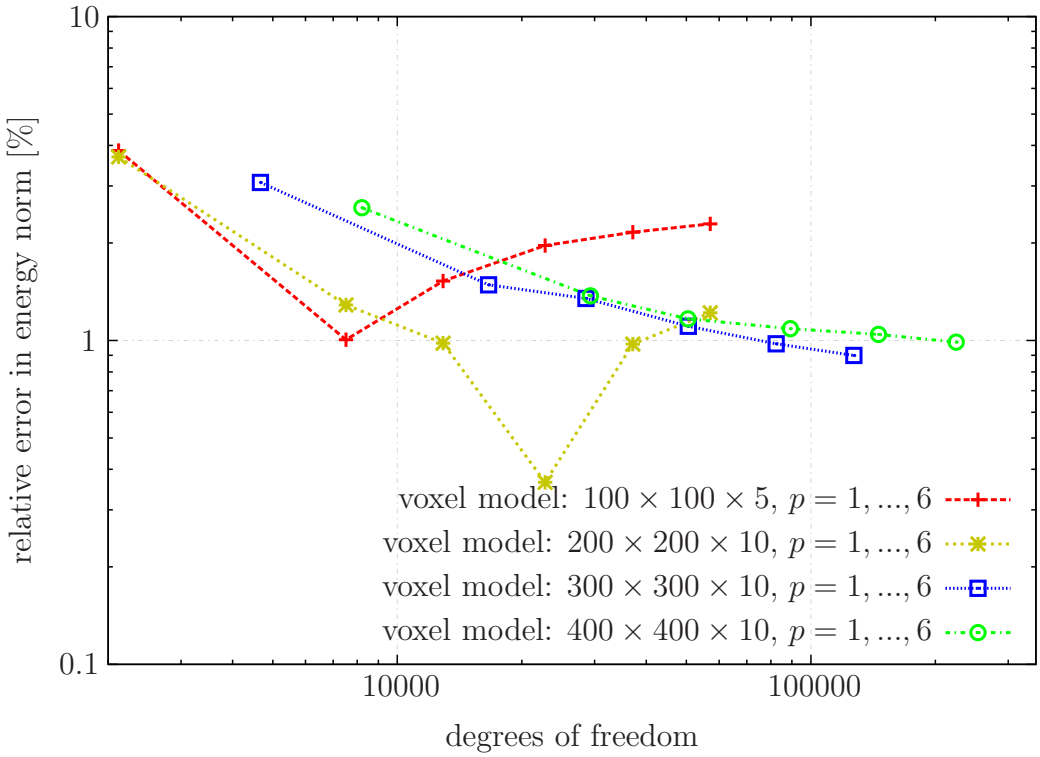


Figure 4.22: Relative errors of strain energy with respect to different voxel models and discretizations

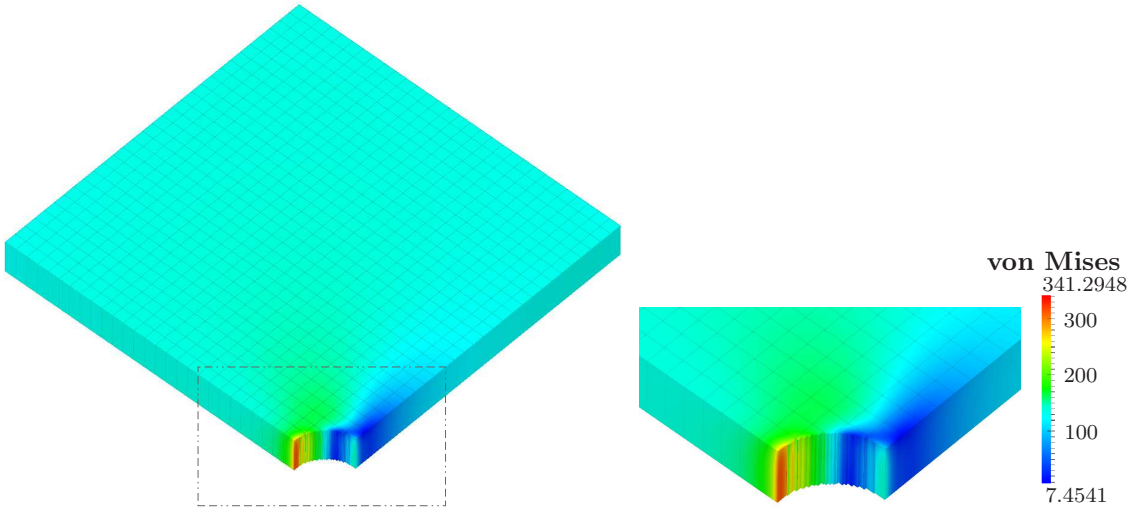


Figure 4.23: Von Mises stress distribution of the $300 \times 300 \times 10$ voxel model, with $p = 4$ and on a mesh $30 \times 30 \times 1$

On the cutline A-B, slight stress oscillations occur adjacent to point A ($x = y = \frac{\sqrt{2}}{2}$). Close to this point, the voxel model $100 \times 100 \times 5$ with $p = 3$ produces an error of around 10%. For finer voxel models with high-order shape functions employed the relative error can be reduced down to below 3%.

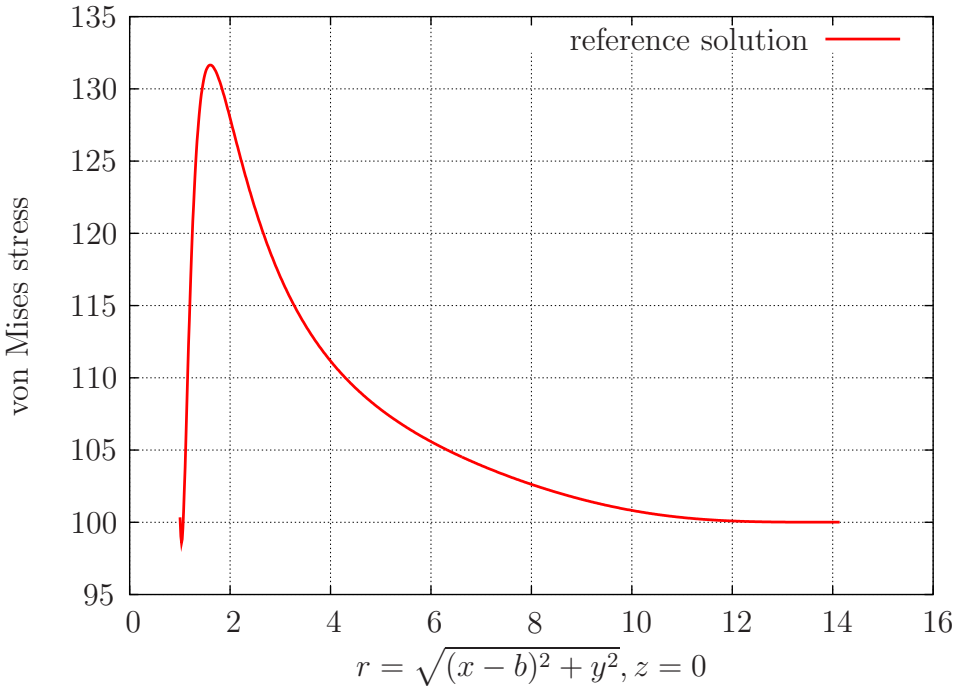


Figure 4.24: Reference solution of von Mises stress along the cutline *A-B*

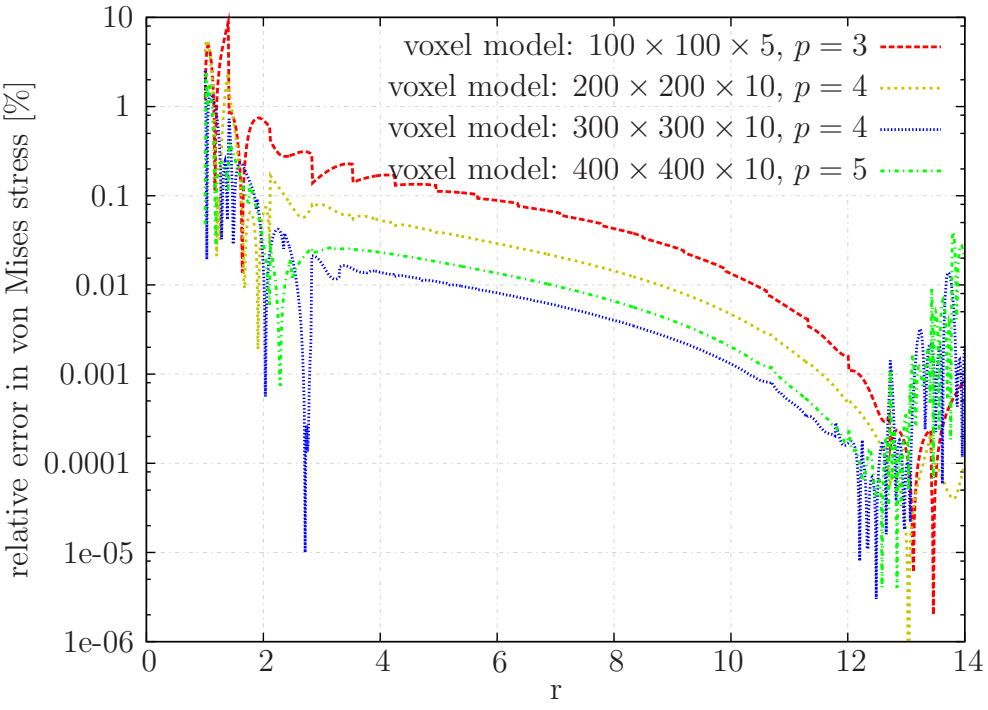
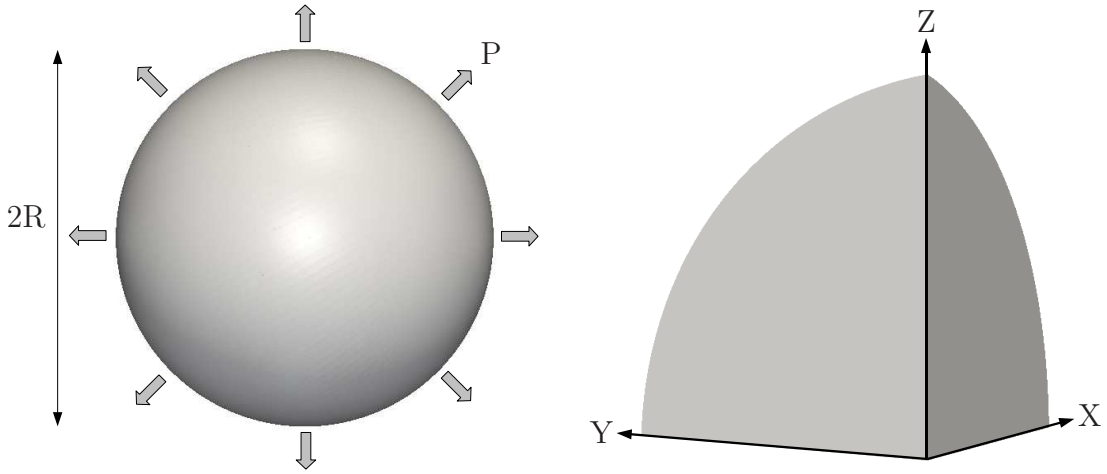


Figure 4.25: Relative errors of von Mises stress along the cutline *A-B*

4.5.3 Pressured homogeneous solid sphere

In this example we investigate the performance of the integration scheme on a spherically symmetric model.



(a) Spherical solid under radial traction (b) One-eighth spherical model

Figure 4.26: Spherical solid model

A spherical solid model of radius $R = 5.0mm$ with uniform traction $P = 1.0N/mm^2$ in the radial direction on its outer surface is plotted in Figure 4.26(a). In the computation only one-eighth of the sphere is modeled due to spherical symmetry, see Figure 4.26(b). Assume that the sphere is a homogeneous isotropic linear elastic solid with Poisson's ratio $\nu = 0.3$ and Young's modulus $E = 1.0$. Due to hydrostatic loading the normal components of stresses are constant and equal to P, while the shear components vanish.

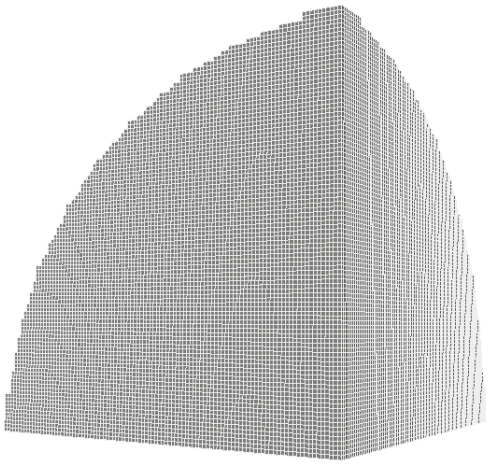


Figure 4.27: Voxel model of the one-eighth sphere with a resolution of $100 \times 100 \times 100$

Assuming the sphere's center fixed, displacement in the radial direction varies linearly from zero to $u_R = \frac{1}{E}(1 - 2\nu)\, P\, R = 2.0$. Three voxel models with resolution $100 \times 100 \times 100$, $130 \times 130 \times 130$ and $160 \times 160 \times 160$ are created for the FCM computation, the coarsest one is illustrated in Figure 4.27. Voxelization of the spherical model generates modeling errors.

As an approximation, these errors are quantified by the relative error in radius evaluated in 2D on the voxels at $x = 0mm$ only. The errors are computed in terms of Equation (4.62), in which $\tilde{R}_i$ is evaluated at $n_s = 500 * n_{edges}$ sampling points equally distributed on the edges of the jagged boundary that approximates the circular arc. The errors are listed in Table 4.3.

Number	**Resolution**	**Error[%]**
1	$100 \times 100 \times 100$	4.232662
2	$130 \times 130 \times 130$	3.313271
3	$160 \times 160 \times 160$	2.627209

Table 4.3: Resolutions and corresponding geometric deviations

From voxel models the finite cell grids are generated, a mesh of $10 \times 10 \times 10$ cells embedding the voxel model $100 \times 100 \times 100$ is shown in Figure 4.28. It is necessary to mention here that since the exact solution is linear, it can be well approximated even when only one cell with $p = 1$ is used [3]. In this example the large number of cells results from implementation details of the fast integration scheme.

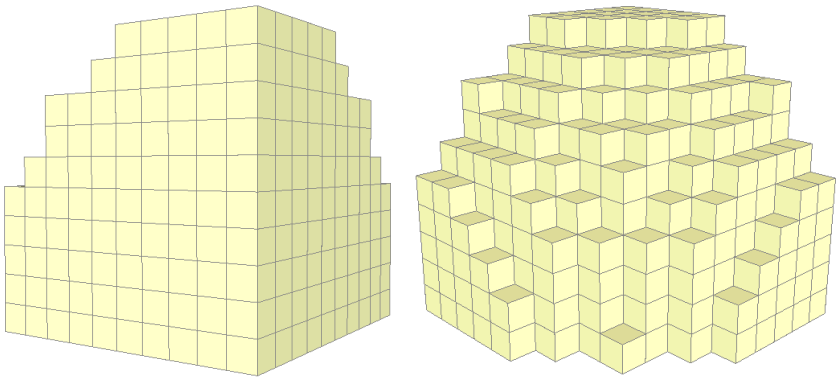


Figure 4.28: Mesh with 630 cells for voxel model $100 \times 100 \times 100$

In the FCM analysis symmetry boundary conditions are applied directly on the cell surfaces, while the Neumann boundary conditions are imposed implicitly on a surface mesh (B-rep) which is non-conforming to the Cartesian grids, see Figure 4.29. The surface of the one-eighth sphere is meshed with 9801 triangles which are sufficient to accurately approximate the exact geometry. The analysis is conducted by a uniform p-extension with $p = 1, ..., 6$ on three voxel models. A contour plot of the radial displacement distribution for voxel model $130 \times 130 \times 130$ and $p = 4$ is shown in Figure 4.30. The exact solution at $R = 5mm$ is $2.0mm$, which confirms a good agreement between the analytical solution and the FCM result.

The accuracy of the integration scheme is further examined by investigating the relative error in von Mises stress, which is defined by

$$e_{vM} = |\frac{\sigma_{vM}}{P}|100[\%] \tag{4.64}$$

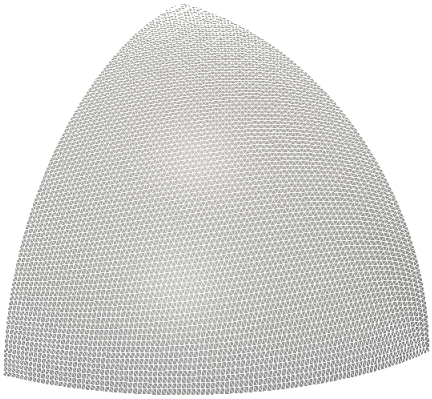


Figure 4.29: A B-rep model of the one-eighth sphere

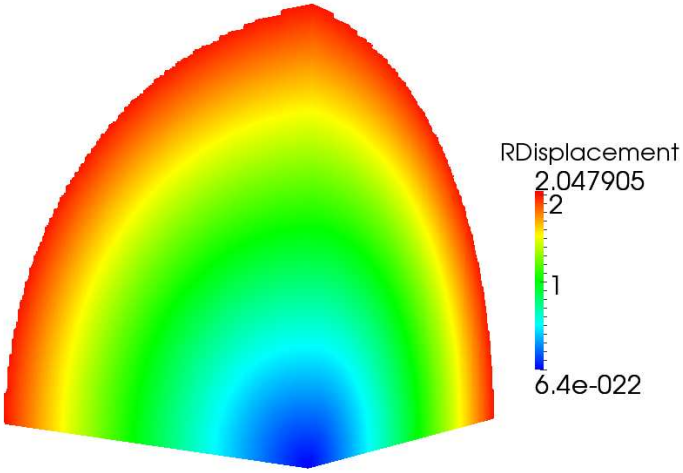


Figure 4.30: Contour plot of the radial displacement (model $130 \times 130 \times 130$, $p = 4$)

Figure 4.31 plots the relative error e_{vM} evaluated on 10000 points equally distributed along the cutline A-B ($x = y = z$) in the interval from 0 to $5/\sqrt{3}$. Similar to the previous example, the modeling errors at the sphere boundary affect the approximation accuracy. Nevertheless, a high accuracy can be obtained when a relatively fine voxel discretization and an appropriate polynomial degree is chosen. In this example relative errors of the three models are all controlled below 3%. Note again, that this error in energy norm is of the same size as the error of geometry as investigated in Table 4.3. In the following example it will be demonstrated that the loss of a small amount of accuracy in stress pays off in gaining incomparable reduction in computational time in CT-based biomechanical simulations.

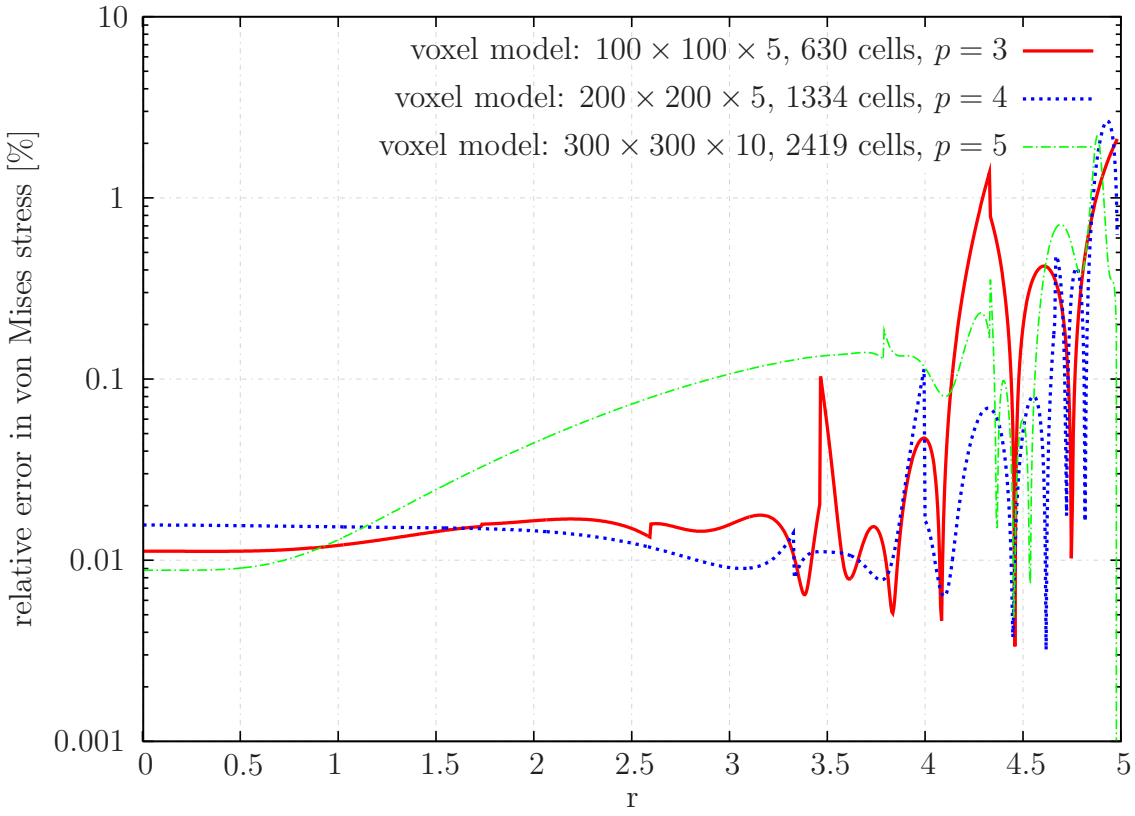


Figure 4.31: Relative errors of von Mises stress along the cutline A-B

4.5.4 Human trabecular bone biopsy

The third example is the numerical analysis of a trabecular bone specimen. A biopsy model is obtained through micro-CT scanning and has been converted to a 3D STL model [91], which can be downloaded from [92], see Figure 4.32.

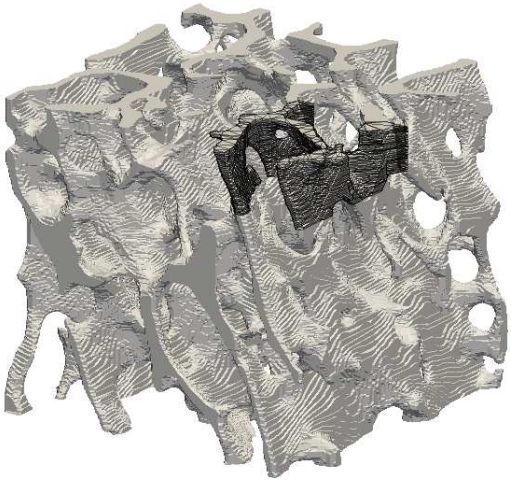


Figure 4.32: A complete human trabecular biopsy model

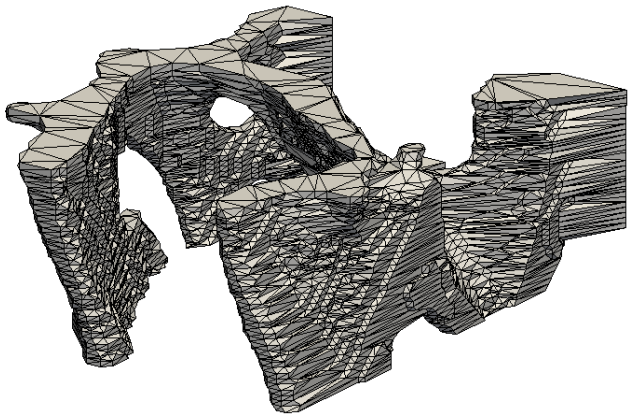
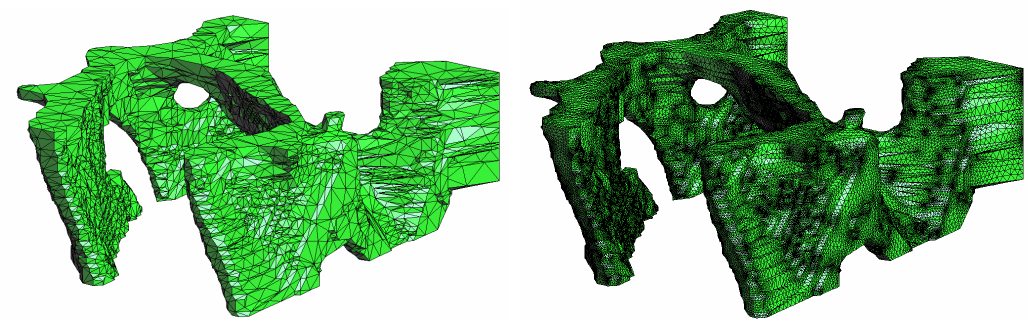


Figure 4.33: A substructure of the human trabecular biopsy model

The goal of this example is to demonstrate the accuracy and efficiency of the fast integration scheme on complex geometric models by comparing it with a FEA result as the reference solution. The convergence study on the model in Figure 4.32 is inhibited by the complexity in geometry and limitation in computer memory. To attain a computable model, a substructure in B-rep (STL format) highlighted in Figure 4.32 has been segmented out and is plotted in 4.33. In this research the new model is regarded to have the exact geometric description, based on which the FEM and the FCM analyses are performed. The material is assumed to be linear elastic with isotropic and homogenous behavior with Young's modulus $E = 1000.00N/mm^2$ and Poisson's ratio $\nu = 0.3$. A prescribed displacement $\mathbf{U_p} = [0,\, 0,\, -0.1]^T$ mm is applied on the top surface of the model while its bottom is completely fixed in three directions. The reaction forces on the topmost surface computed by the two methods will be utilized for comparison.

A series of FEA computations has been conducted to generate a reference solution. The mesh generator NETGEN [93] is adopted as the preprocessor to discretize the STL model and to generate 10-node tetrahedral meshes. Different levels of discretization are created. Figure 4.34 displays the coarsest and the finest mesh with 23,450 and 3,437,623 elements, respectively. The mesh generation is a computationally expensive process for finer discretizations. The meshing time is listed in Table 4.4.



(a) Coarsest tetrahedral mesh (b) Finest tetrahedral mesh

Figure 4.34: Model discretizations using NETGEN

The commercial FEA software package ABAQUS [94] serves as both solver and postprocessor. With boundary conditions applied the FEA results are computed on each discretization. The computational time measured on an AMD Opteron 250, 2.4Hz machine (with one core activated) is listed in Table 4.4, in which the summation of mesh generation and computation time is calculated and tabulated in the last column.

	Number of	**Computational time (s)**		
	Elements	Meshing	Computation	Total
1	23450	405	23	428
2	37050	388	43	431
3	81206	482	1501	633
4	300611	2300	1006	3306
5	480679	7516	2160	9676
6	860028	2358	6533	8891
7	1577863	8159	14410	22569
8	3437623	12083	43431	55514

Table 4.4: FEM computational time of the human biopsy model using NETGEN and ABAQUS measured on an AMD Opteron250, 2.4Hz machine with one core activated

Figure 4.35 shows the convergence of the reaction force F_z caused by the prescribed displacement on the topmost surface of the model under h-refinement of the mesh.The converged value of $\mathbf{-2470}$ is taken as the reference solution for the FCM comparison.

The FCM allows a fast import of a voxel model derived from a CT scan which can be directly implemented into an analysis procedure. In this example, however, such a CT scan is not available. One possible way of retrieving the voxel dataset is to voxelize the STL model with different resolutions in a CT-like fashion. The octree-based voxelization program mentioned in Section 4.4.1.2 is utilized to generate voxel models with various resolutions. For comparison purpose, four voxel models with resolutions of $81 \times 75 \times 30$, $135 \times 125 \times 50$, $270 \times 250 \times 100$ and $297 \times 275 \times 110$ are generated. Figure 4.36(a) depicts one voxel model with a resolution of $270 \times 250 \times 100$.

This model, with the setting of $10 \times 10 \times 10$ voxels per cell, is discretized into 6750 cells. Cells that lie completely outside the physical domain are discarded for saving computational cost, and the remaining 2124 cells are plotted in Figure 4.36(b). With the same boundary conditions applied as in the FEM analysis, numerical results computed on these four voxel models are obtained by a uniform p-extension. Figure 4.37 presents the displacement distribution in z-direction on the voxel model $270 \times 250 \times 100$ on mesh $27 \times 25 \times 10$ with $p = 4$.

The convergence curves of reaction forces on the four voxel models are plotted in Figure 4.38. The setting of $5 \times 5 \times 5$ voxels per cell is adopted for the two coarser models, while $10 \times 10 \times 10$

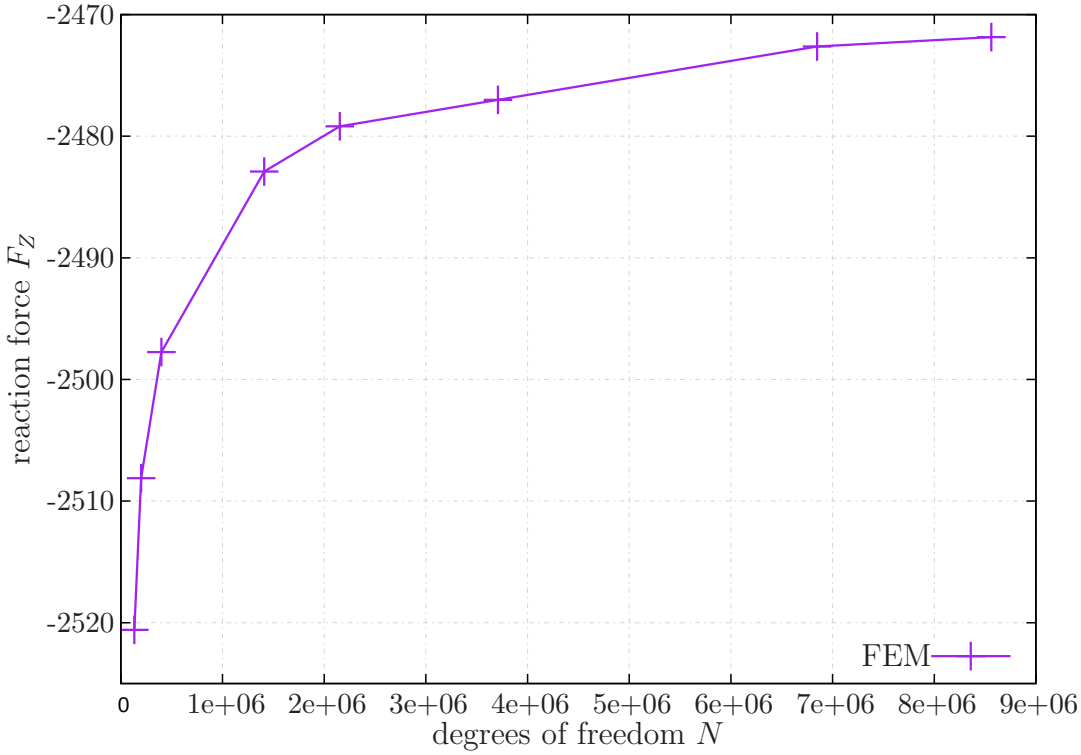
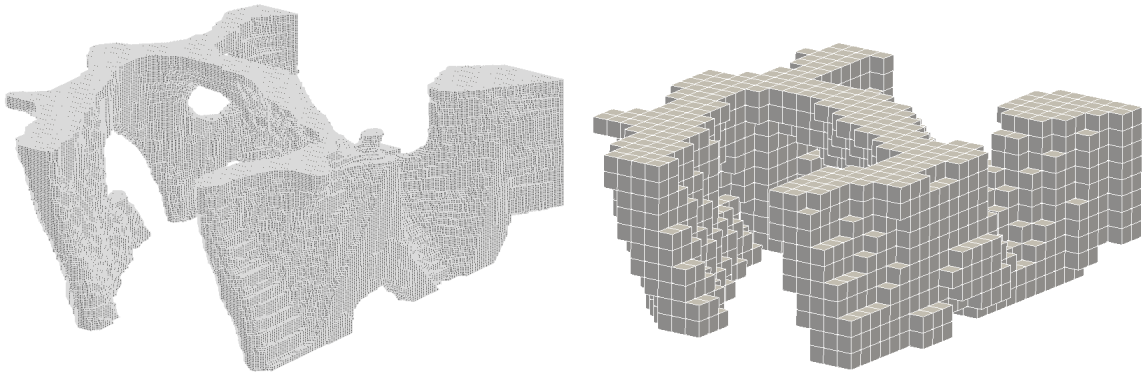


Figure 4.35: Convergence of the reaction force (FEA)



(a) Voxel model with a resolution of $270 \times 250 \times 100$

(b) FCM discretization

Figure 4.36: A voxel model and its FCM discretization of the trabecular biopsy specimen

voxels per cell is chosen for the two finer ones. On the model with the lowest resolution the curve converges slowly, since the modeling error of this voxel model is relatively large. As the voxel model is refined, the FCM results get closer and closer to the reference solution. The FCM results of the last two finer models converge to approximately $\mathbf{-2430}$ which produces a relative error of below 2%.

As mentioned at the end of the second example, the computational speed is the highlight of this scheme. The FCM code is parallelized using OpenMP technique, and was executed on an

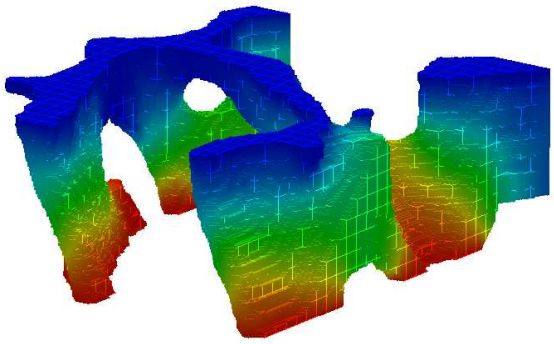


Figure 4.37: Displacement in z-direction

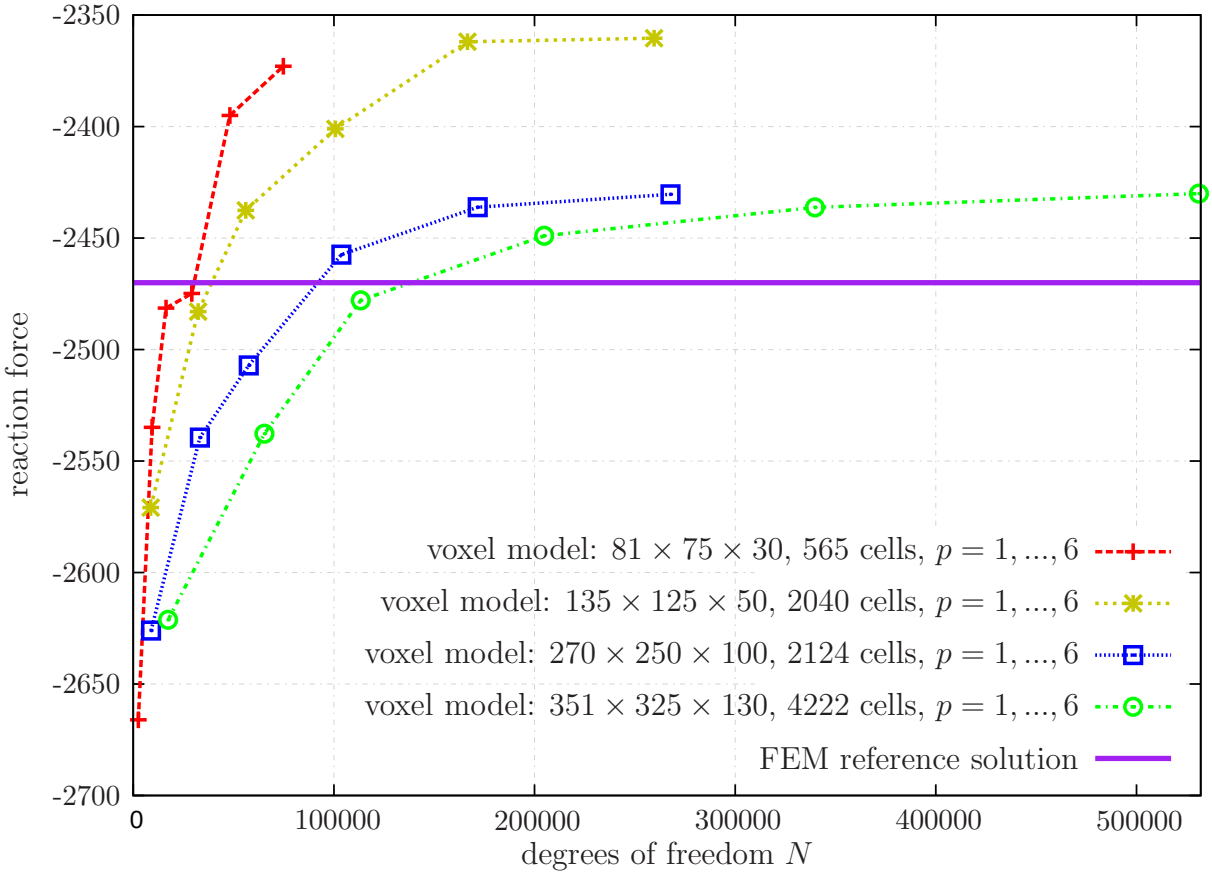


Figure 4.38: Reaction force convergence curve of the FCM results

Intel Xeon W5590, 3.33GHz work station, 8 cores. For the reason of fairness in comparison with the FEM computation, only one core is activated. The computational time was measured on the voxel model $270 \times 250 \times 100$ which produces plausible results according to Figure 4.38. Table 4.5 lists the computational time of this model for $p = 1, ..., 6$.

p	DOF	Computational time (s)				
		Reading	Stiffness	Assembly	Solver	Total
1	8913	0.06	0.23	0.04	0.15	7.02
2	33264	0.31	1.25	0.51	1.79	15.45
3	57615	0.71	3.20	1.63	5.81	28.55
4	103755	1.59	7.53	6.24	18.39	62.55
5	171684	3.31	16.99	19.09	51.22	138.63
6	267774	6.98	33.10	52.35	130.91	299.47

Table 4.5: Computational time of the $270 \times 250 \times 100$ model with FCM parallelized with one core

The total computational time in the last column includes all steps of one analysis, from reading in the voxel model to post-processing. The time for FCM to get a converged result is about 300s when $p = 6$, while the FEM computation time for the coarsest tetrahedral model is 428s, see Table 4.4. For fairness the 428s is divided by the CPU frequency ratio of two computers (3.33HZ/2.4HZ=1.3875) and equals to 308s, which is almost the same as the FCM computation that produces highly accurate results. It's also noteworthy that besides the FEM computational time listed in Table 4.4, a significant time is required in an a-prior preprocessing step to convert the CT scan into a B-rep model based on which tetrahedral meshes are generated. This conversion is labor-intensive. Note that this tedious step is avoided in the FCM with which the CT voxel model can be directly processed with little effort.

For investigating the efficiency of parallelization, the same computations were performed on the same machine parallelized with 8 cores. The new computational time is tabulated in Table 4.6.

p	Stiffness matrix			Assembly	Solver	Total (s)
	1 core	8 cores	efficiency [%]	8 cores	8 cores	8 cores
1	0.23	0.10	28.75	0.04	0.13	4.16
2	1.25	0.29	53.88	0.46	1.18	6.75
3	3.20	0.46	86.96	1.50	3.13	11.34
4	7.53	1.32	71.31	5.95	8.05	25.28
5	16.99	2.57	82.64	17.82	17.92	58.18
6	33.10	5.50	75.23	50.21	50.25	145.01

Table 4.6: Computational time of the $270 \times 250 \times 100$ model with FCM parallelized with 8 cores

The efficiency of parallelization is estimated by dividing the speedup by the number of cores. In this example, the parallelization efficiency has been calculated based on the stiffness matrix computational time measured in the two contexts tabulated in Table 4.6. The highest parallelization efficiency is approximately 87% for $p = 3$.

The high efficiency in the FCM computation is contributed from both the preprocessing and the fast integration. In this example, the preprocessing time (cells generation) takes approximately 0.1 second, which is of big advantage when comparing with the processing time of the FEM computations. Furthermore, the fast integration scheme drastically shortens the stiffness matrix computational time, which in this problem takes less than 5% of the total time after parallelization. The main part of the computational time is dedicated to the solver, whose performance on solving the linear system of equations plays an important role. In the FCM applications, iterative solvers can be applied to solve small-sized problems with relatively simple geometries, while fast and reliable direct solvers are more commonly used to handle large linear system of equations resulted from geometrically complicated or inhomogeneous models due to the bad conditioning of equation systems.

Chapter 5

Computational steering for orthopaedics using the FCM with fast integration

5.1 Introduction to hip replacement

The hip is a joint where the femur meets the pelvis. It consists of two parts, the femoral head which is covered by a layer of smooth cartilage and the acetabulum which is part of the pelvic bone. There is fluid flowing in between the two parts helping the ball-socket-like joint move smoothly, see Figure 5.1.

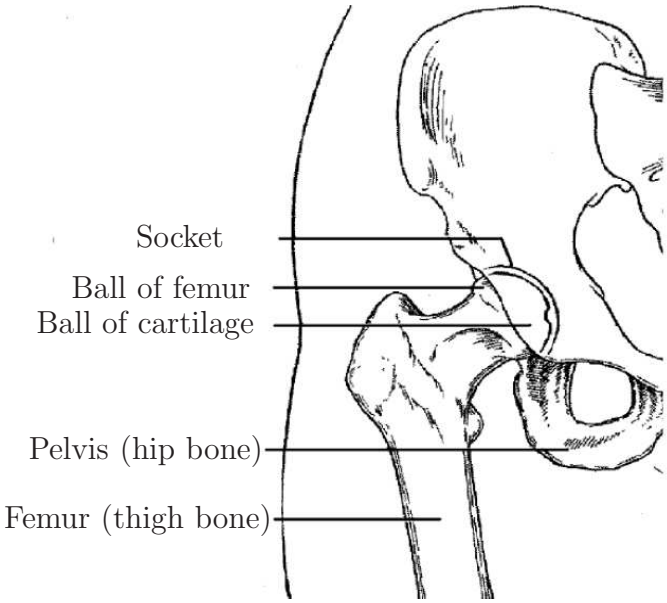


Figure 5.1: Hip joint [95]

The cartilage can be worn down by a degenerative joint disease called osteoarthritis, aging or injury. The breakdown of the cartilage roughens the contact surface and thus causes pain and eventual loss in joint movement. In this case the diseased or damaged parts of the hip joint

must be removed and replaced by an artificial joint which has smooth surfaces. This surgery is called the total hip replacement (THR).

The THR is an effective treatment for severe hip joint damages and diseases. There are approximately 1 million THR operations carried out worldwide per year [96] with a success rate over 90%. The majority of patients are in their 60s or 70s. In THR the femur's head along with the surface layer of the socket are removed and replaced with a metal ball attached to a metal stem fitted into the shaft of the femur. The metallic ball and stem are commonly called the "prosthesis" or "implant". Additionally the damaged socket in the pelvis is replaced by a plastic or metal socket. Prosthesis can be categorized into two types, cemented prosthesis and non-cemented prosthesis, see Figure 5.2. Upon inserting a cemented prosthesis which has smooth surfaces into the femur shaft, it must be fixed in place with a special cement, which accommodates uniform load transfer between the prosthesis and the irregular texture of bone. As an alternative, non-cemented prosthesis which has a microscopic porous surface that provokes ingrowth of bone material into the prosthesis stem is often used for an improved fixation especially for young patients.

(*a*) A cemented prosthesis (*b*) A non-cemented prosthesis

Figure 5.2: Two types of prostheses [97]

The development and application of THR have tremendously improved the quality of life for the majority of patients who received the operation. However, after a THR operation, the muscle and joint loads are carried almost entirely by the prosthesis stem, while the remaining part of the femur exhibits stress shielding [98]. Consequently, most post-operative femurs undergo long-term changes in their bone architecture, because the bone of a healthy human being or animal will remodel in response to the loads it is subjected to, according to Wolff's law.

These changes include bone resorption in the cortical region and formation of a fibrous tissue between the implant and bone interface. The resorption and formation increase with time and will cause cortical thinning, reduction in bone density and loss of integrity of the implant-bone interface (aseptic loosening). As a result, patient suffer from pain and functional restrictions, eventually a revision surgery might be unavoidable. Through retrospective evaluations based on the long-term hip replacement registration data from the Nordic Hip Registers, the implant design, surgical technique and patient age are found to be the three main risk factors in a THR [99]. Despite of this database it is yet unclear of how the three factors precisely affect a THR.

Finite element based simulation methods have been proved to be accurate and efficient tools for biomechanical analysis [1, 22]. For many years FE simulations have been carried out to assist surgeons and implant designers in improving their understanding of the origin and patterns of the stresses and strains involved, and how a failure is formed [100, 101, 102]. Furthermore, it is used for preclinical testing and design testing of new prostheses [103, 104]. To the author's knowledge, these tests are mainly targeted at validating a certain prosthesis design, so far there's no real-time interactive surgical planning system that recruits patient-specific data according to the individual mechanical properties of the involved bone. The rapid development of high performance computers and new simulation methods enhances the possibilities of establishing such a system which, if innovatively constructed, can be recruited to perform preclinical tests for specific patients in real-time via computational steering.

5.2 Introduction to computational steering

During the past decades numerical simulations have been more frequently used to obtain numerical solutions to the differential equations which can not be solved analytically. From the numerical results of the simulation, knowledge of background processes and physical understanding of the simulation region can be obtained. Traditionally the simulations are run in batch-mode, after which visualization tools are used to post-process and visualize the dataset created by computation. For this reason the interaction between simulation and visualization is a long process with limited user interaction.

The recent advances in computing power and computational technology now enables a close coupling between simulation and visualization to a degree where the influence of changes in input parameters can rapidly be computed and visualized. This way of scientific visualization is called computational steering (CS). A good computational steering tool can help the user to find optimal configurations by dynamically modifying computations while they are running and even steer the computations in real time.

Primarily there are three requirements for the establishment of such a computational steering tool. First, because in most real-world examples the data are vast, efficient interactive visualization techniques for large data sets are needed. Second, to limit the computational time of a large-scale simulation to the permittable range of a real-time simulation, high performance simulation kernels play a key role [105]. Third, if the simulation and the visualization are running on various sources such as supercomputer back-ends and visualization front-ends [106],

high bandwidth and low latency networks are called for to handle the large amount of data to be transferred between simulation and visualization kernels interactively.

The basic idea of computational steering is straightforward; however, its implementation is tedious and labor-intensive. The establishment of a new CS system often requires many modifications on the original scientific code. Three broad approaches have been addressed in [107], they are program instrumentation, direct scientific computation and recasting scientific computations respectively. In program instrumentation an existing scientific application is employed and only small modifications of the source codes are made to provide access points for parameters and results. In direct scientific computation a code is broken up into various modules to allow users to interact with them. In recasting scientific computations a simulation is constructed using reusable computational components to be connected with a visual programming environment. For the above three approaches the effort of modifying the original scientific code increases; however, in turn more flexibility and interactivity is available.

5.3 Computational Steering for Orthopaedics

5.3.1 Motivation

Total hip replacement is a successful treatment for joint failures caused by bone diseases or accidents. The endurance of a THR surgery is determined by prosthetic, surgical and patient factors. Correct selection of a patient-specific prosthesis can prominently improve the surgery quality and postpone or even avoid revision. Hence, preoperative planning is an essential ingredient for a successful surgery.

In most clinical centers worldwide the preoperative selection of a prosthesis is done with a template of implants drawn on a translucent sheet and an anteroposterior (AP) X-ray of the patient's hip joint [108, 109]. The prosthesis outlines are superimposed onto the X-ray transparency to decide whether a prosthesis is geometrically suitable or not. The drawbacks of this approach are that only two-dimensional data are available while the three-dimensional information is still missing, also the rotational misalignment is not controlled and the revision on the prosthesis position is only limited in the coronal plane. Thus, which prosthesis is chosen is decided by a surgeon's appraisement based on his subjective judgments from intraoperative experiences, a rigorous method is still lacked. In recent years, new Computer Aided Orthopaedic Surgery systems, e.g. HipNav systems [110], [111], have been developed. These tools aim at visualizing the prostheses and patient's femur in 3D and providing surgeons the positioning and geometric information to assist surgeons in selecting a best-fitting prosthesis in a more precise and reproducible manner. From the geometric perspective the implementation of these tools in an orthopaedic surgical planning ensures a good preoperative selection of the implant, which from the mechanical perspective may be inappropriate. On top of the geometric information the patient-specific biomechanical information, e.g. physiological load transmission from the implant to the proximal femur, is more crucial concerning the long-term stability and longevity of an implant. If an ideal implant is selected the post-operative strain patterns should replicate the preoperative physiological strain state so that the stress shielding can be prevented.

This aspect can be dealt with finite element based simulation methods which have been proved to be accurate and efficient tools for biomechanical analysis [1, 22]. They have been carried out to assist surgeons and implant designers in improving their understanding of the origin and patterns of the stresses and strains involved, and how a failure is formed [100, 101, 102]. Furthermore, it is used for preclinical testing and design testing of new prostheses [103, 104]. These approaches, however, are mainly targeted at validating a certain prosthesis design, not at selecting the implant for a specific patient prior to surgery. Moreover, such studies are time-consuming and not yet suited for patient-specific surgery planning.

Ideally, a computational tool for patient-specific surgery planning should include the following features:

- to employ patient-specific data, such as femur geometry and material information
- short preprocessing time
- to perform fast and accurate computation of stresses and strain states
- to provide user interaction and direct feedback for computations of multiple scenarios

In the following, a methodology which is capable of fulfilling these requirements will be presented. It was implemented and tested not on a high-end system but standard hardware at a current price of approximately 5000 Euros. This part of the dissertation is structured as follow: the surgical planning system is firstly overviewed in Section 5.3.2, in Section 5.3.3 the structure of the computational steering system will be presented to give an overview of the system establishment, afterwards the visualization approaches are discussed in Section 5.3.4, Section 5.3.5 details the simulation method, the coupling is described in Section 5.3.6, Section 5.3.7 gives a demonstration example of the computational steering system, whose accuracy was validated by a biomechanical example in Section 4.5.

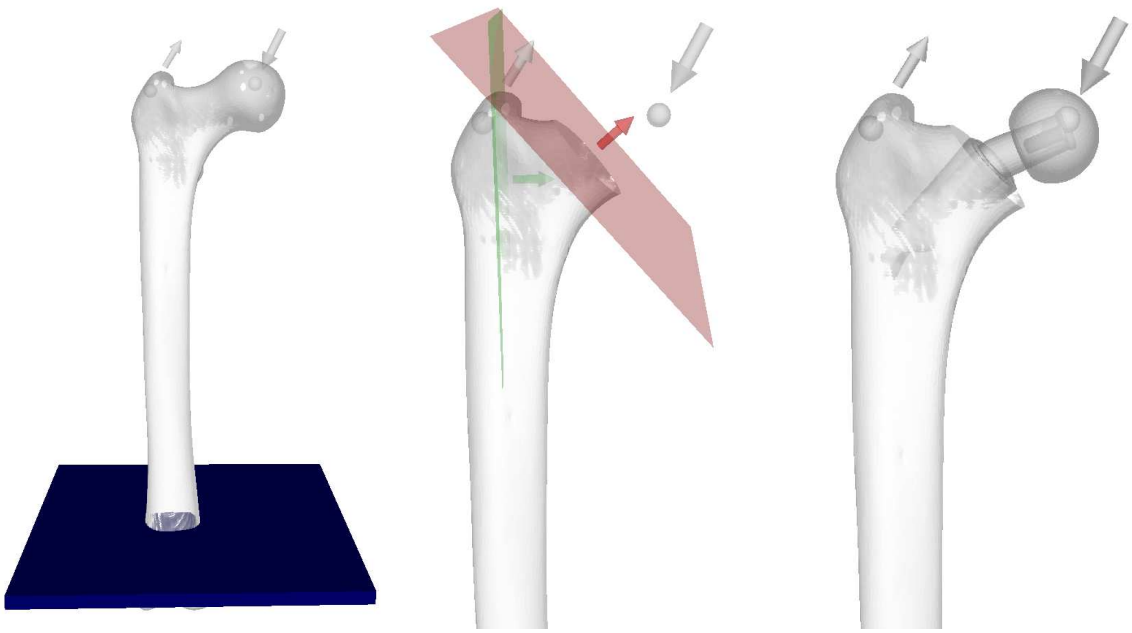
5.3.2 Surgical planning system overview

This planning system employs patient-specific bone information and can provide patient-specific biomechanical information in real-time. The system was constructed as a collaborative work with Dick et al. who developed the user interface as well as the visualization kernel of the system [112, 113].

In a preprocessing step prior to the starting of the planning system, pre-operative CT scans are segmented and processed in a separate program to provide a clean voxel model as well as a surface description of the geometry (B-rep model). This B-rep geometry is mainly used as a reference for application of boundary conditions and visualization of stresses on the femur surface.

The system starts from importing the segmented CT data and the B-rep model. Users can rotate and zoom in/out of the femur. In addition, users have the option of viewing details hidden

in the interior of the bone through a lens-like focus and context metaphor. The material law relating the CT Hounsfield Unit value with the bone elasticity modulus can be input by the user. Starting with the segmented CT voxel model at the initial CT resolution, a hierarchical octree representation is constructed in a bottom-up order. Property of a voxel at a coarser resolution level is average of the properties of all voxels contained at a finer level of this voxel. Once the hierarchy has been constructed, the user first selects the resolution level at which to perform the simulation. To apply the boundary conditions, a movable plane indicating at which place the bone is clamped and two loads with adjustable magnitude, position and direction are available to simulate the realistic loading conditions, see Figure 5.3(a). As a preclinical study, the physiological stress distribution in the femur can be simulated in this step before the implantation simulation takes place.



(a) Femur model with fixation and loads (b) Femur model segmented with two cutting planes (c) Femur model with an implant model inserted

Figure 5.3: Surgical planning system overview

Two movable cutting planes are provided to simulate the femur tissue removal process, see 5.3(b). To validate different testing prostheses via computational steering, the program runs in an iterative loop where one iteration is one completely new simulation. Implant candidates are represented by B-rep models, which can be translated or rotated in 3D. Each implant can freely be placed anywhere in the computational domain including inside the femur to test the post-operative stress states, see Figure 5.3(c).

When an implant is placed inside the femur, the relative motion between the implant surface and the femur's interior structure should be taken into account. Simulation of the relative motion involves the modeling of a frictional contact of the bone-implant interface, which complicates the establishment of the planning system. As an approximation, the implant is considered to be perfectly bonded to the femur at the places where two objects are in contact

or overlapping.

5.3.3 General procedure and system setup

Since the "Computational Steering" concept was proposed a decade ago, many relevant approaches and applications have been developed [81, 106, 114, 115]. The general approach in constructing a computational steering system is to separate the visualization and simulation into two communicating processes, possibly running on different processors. Traditionally the simulation is run in parallel on a high performance computing (HPC) cluster using the Message Passing Interface library (MPI), while the front-end visualization system is allowed to connect and disconnect as required from a long running simulation and to take a service-oriented approach to visualization and steering. Relying on the HPC clusters, the computational load can be distributed to a number of processors in order to keep the simulation time within a permissible range. However, such clusters are not easily affordable for medium or small institutes or companies. As the development of multi-core processors and graphics processing units (GPUs), high performance computing is no longer only available to computer scientists in research labs. HPC systems are now being built from the same commodity chips found in desktop personal workstations [116]. The affordability of the new HPC system, which consists only of one computer with multi-core processors and GPUs, has remarkably increased lately. Performing simulation on a single computer is then made possible.

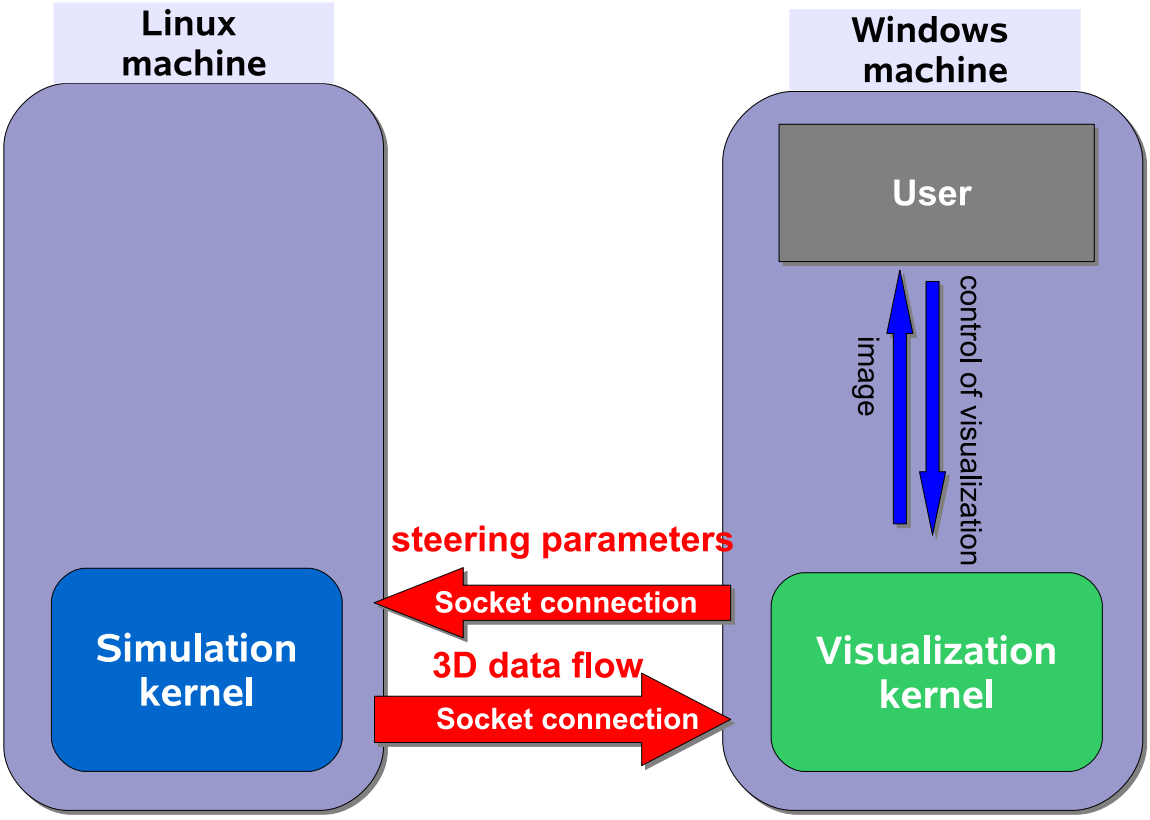


Figure 5.4: Computational steering pipeline

The computational steering system for surgical planning is established with two machines, one for simulation, one for visualization. The simulation was implemented in C language under Linux ultilizing the finite element code *AdhoC* [117, 118, 87] and compiled on a commercially available workstation (Dell Precision T5500, 24GB Memory, Intel Xeon W5590 CPU, 3.33GHz, 8 cores). The visualization was implemented by Dick et al. in Visual C++ under Windows on the other workstation with a powerful graphics card (NVDIA Quadro FX 5800 graphics card with 4GB memory hosted in a Windows 7 machine with 4GB RAM, Intel Xeon X5550 CPU, 2.67GHz, 8 cores). In order to foster a strong coupling of two codes running under different system architectures, the communication is enforced through high-speed socket connection with speed 1Gbps. The general setup procedure is depicted in Figure 5.4.

Data transfer via network in between the two machines causes an additional latency, which decreases with the amount of data transferred. A dedicated data reduction scheme has been developed for efficient data communication between the two terminals. This scheme will be addressed in Section 5.3.6.

5.3.4 Visualization techniques

The purpose of visualization in computational steering is to provide a real-time visual representation of the state of a simulation. To provide an intelligible representation and a rapid response for the visualization, efficient rendering techniques and effective visualization patterns have been developed by Dick et al. [112, 113]. The visualization pipeline is depicted in Figure 5.5.

On the visualization machine both CPU and GPU are working for the visualization kernel. In the first run, the segmented data and the B-rep geometry are read in by the visualization kernel. Voxel data of the segmented femur are converted into a uniform format and are sent to the simulation kernel, which computes the stress results and transfers them to the visualization machine. This procedure is carried out on the CPU for once only to set up the system, after which the system enters an iterative loop containing the following two steps:

1. The GPU instead of the CPU is utilized to rapidly process the results and render the image. Three visualization patterns depicted in Figure 5.6 have been developed for different purposes.

In Figure 5.6(a) the femur and implant meshes are rendered as semi-transparent surfaces and the von Mises stresses are rendered in red through volume ray-casting. The magnitude of stress is indicated by the intensity of the red color. This visualization pattern tends to provide the surgeon some basic information about the stress distributions and a general overview of the load transfer in the interior of the femur. In Figure 5.6(b) a more advanced technique which combines volume rendering with transparent, shaded, and antialiased lines is employed to indicate stress directions computed from a 3D stress tensor field. The directional information provided by these lines gives a visualization of the flow of forces through the bone as indicated in Figure 2.7. Figure 5.6(c) illustrates the third visualization pattern − the comparative stress visualization, which shows the differences between the simulated stress tensor field resulting from a virtual implant surgery and the physiological stress distribution. Note that the loading conditions (directions, positions and magnitudes) are completely the same for

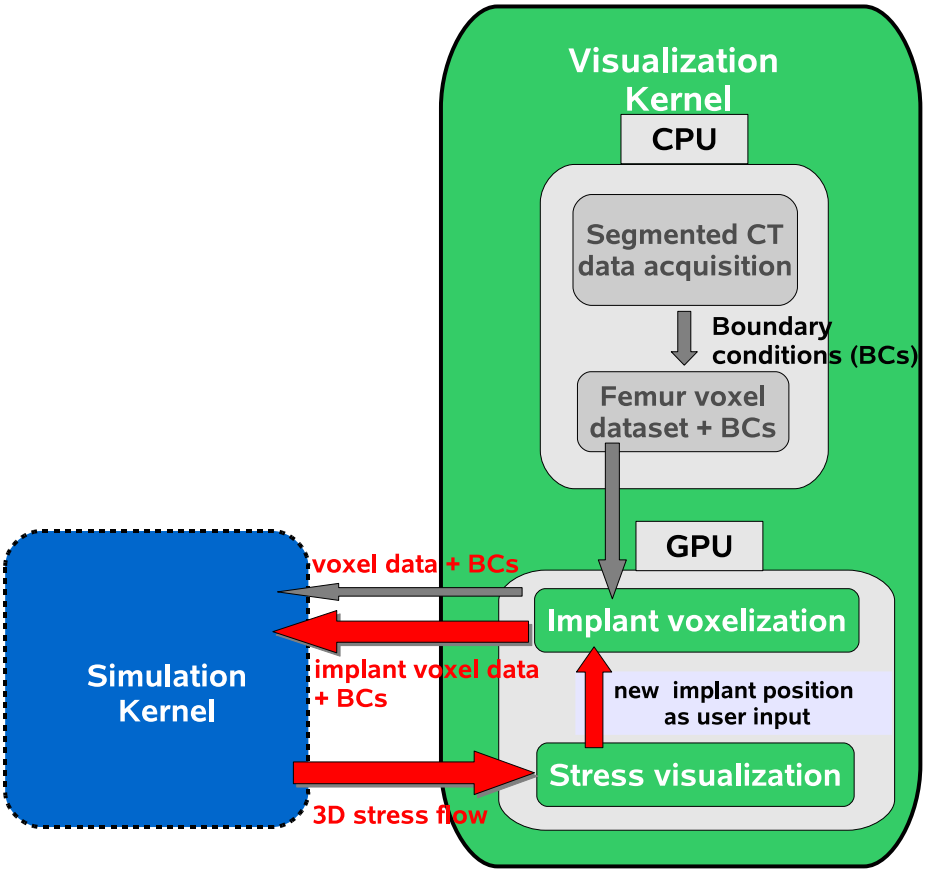
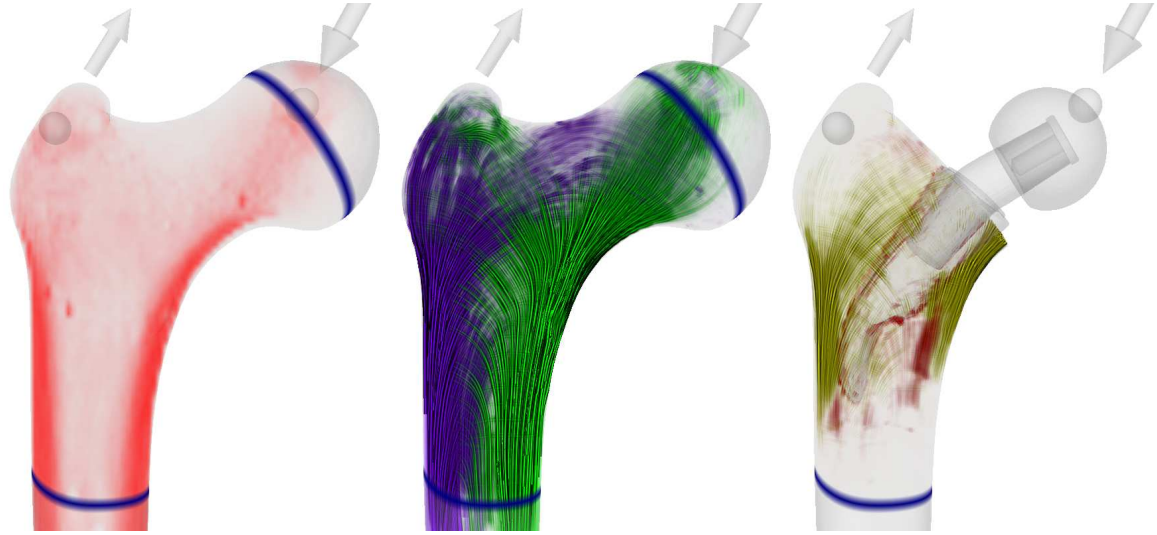


Figure 5.5: Visualization pipeline



(a) Volume rendering of von Mises stresses in red color via ray casting

(b) 3D stress tensor fields visualization by stress lines

(c) 3D comparative stress visualization by stress lines (normal components)

Figure 5.6: Visualization patterns

the two states in order to ensure a meaningful stress visualization. Principal stress directions in the physiological state are utilized as a reference frame, on which the two stress tensor fields are decomposed into normal and shear stress components at each point and the differences are visualized with respect to these components. The visualization of changes in the normal stress components can be used to identify the regions of stress shielding according to the Wolff's law, while the visualization of changes in shear stress components at the implant-femur interface can be employed to help to judge the initial rotational stability[1] which must be secured for the integration of a non-cemented implant to be sustained over the long term [119].

2. As a simplification, the implant model is generated as a new voxel model which has the same voxel dimensions as the CT data. The voxel data are acquired by voxelization of a B-rep implant model through an oct-tree based fast voxelization technique. This step is categorized to the visualization, not the simulation for two reasons. First, by implementing a GPU-based voxelization approach, the execution time is much shorter than on the CPU. Second, the implant voxel model which is stored in the GPU can be directly employed and visualized in the rendering process. During the steering when the implant position is modified by the user, a new position information of these voxels is sent to the simulation kernel to trigger a new computation.

The GPU-based visualization techniques enable the visualization kernel to render large-scale data in the range of milliseconds. As a result, the main computational effort in the current computational steering system is devoted to the simulation.

5.3.5 Simulation method

In the mechanical analysis of human bone, the finite element method proves to be an effective tool for stress analysis and failure prediction. Two types of model construction and mesh generation methods, "voxel-based" and "structure based" methods, are commonly used to predict the bone mechanical response [1], see Section 2.4.1.

The "structure based" method shows higher accuracy while the "voxel based" method is more efficient concerning the computational time. The drawback of the "structure based" method is the computational efficiency which is hindered by the preprocessing step in which a geometrical model with smooth surface description is first constructed from CT-derived voxel dataset and then a 3D mesh is generated from the geometrical model. The "voxel based" method is comparatively advantageous with respect to model creation and mesh generation; however, its efficiency is undermined by having to solve the system of equations resulting from the vast number of 8-noded elements converted directly from voxels, when a good accuracy is demanded. A "voxel-based" FEM simulation system was developed by Dick et al. [112]. This system shows a good efficiency but a limited accuracy as it incorporates only linear elements and works with voxel models with low resolutions.

The FCM with fast integration is an efficient and reliable alternative to the two choices above.

[1]Rotational stability means that the stem is resistant to the articular forces that induce rotation around the implant's longitudinal axis

Application of this method in the femur analysis has three main advantages: first, the voxel model obtained from a CT scan can be used as an immediate basis for the computation without segmentation or complicated mesh generation. Second, it is possible to speed up the computation of cell stiffness matrices such that it takes only a small fraction of total computational time. Third, less time is needed in the solution phase in comparison with the general "voxel based" methods in which linear shape functions (h-version FEM) are used, due to the fact that p-version models generally have less degrees of freedom than h-version models. With FCM applied to the computational steering system, both high efficiency and high accuracy can be achieved, thus the real-time simulation is made possible.

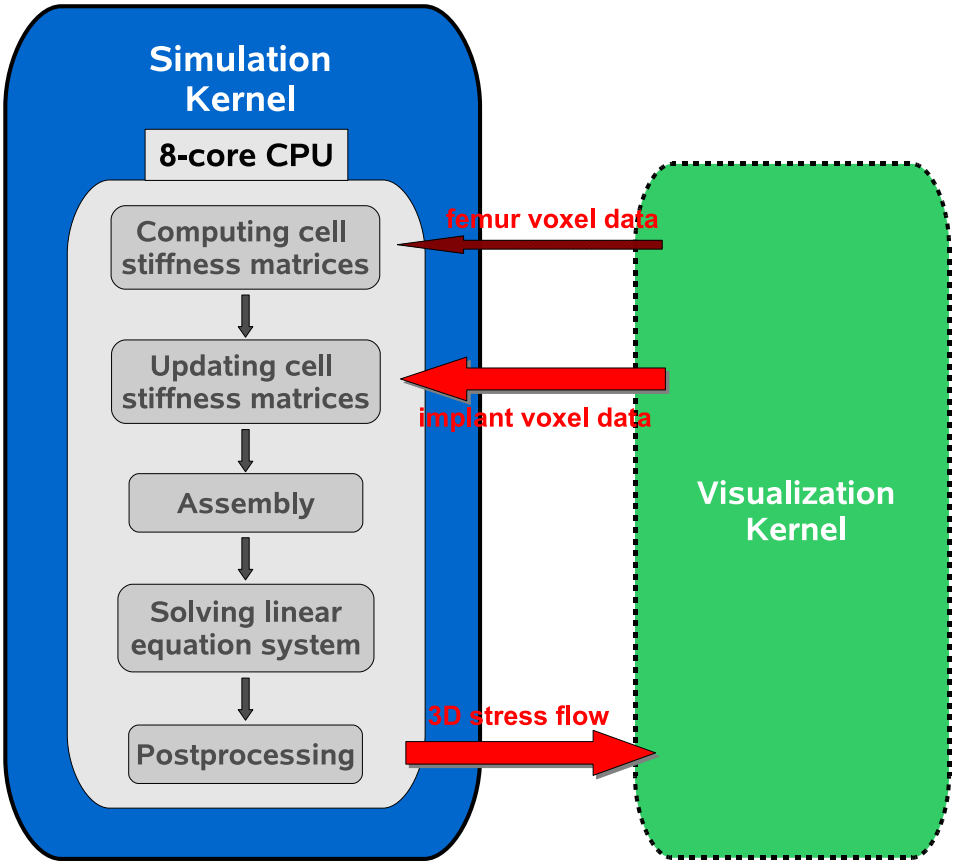


Figure 5.7: Simulation pipeline

Figure 5.7 depicts the simulation pipeline. The polynomial degree p and the number of voxels in each direction per cell n_v are user-defined parameters and are independent of the visualization kernel. After receiving the voxel model, the simulation kernel generates a rectangular domain which embeds the entire femur. This domain is subdivided into a finite number of equal-sized grids (cells). To account for different implantation scenarios that might happen during the steering process, a large fictitious domain which has the capacity to enclose both femur and implant is needed. Considering the real operation scenario, the most essential stress state for the surgeon is when the prosthesis shaft is completely embedded by the femur. Hence, other transition states during the steering, e.g. when the implant is partially in contact with the femur, are of much less importance and can be computed with lower accuracy. With these

considerations, the computational domain is only confined to embed the initial geometry of the femur. Cells lying completely outside the femur's physical domain are discarded to save computational cost.

In the initial run, stiffness matrices of all cells are computed altogether by implementing matrix-matrix multiplications using the BLAS subroutine provided by the Intel MKL. The computed stiffness matrix of each cell is stored in the memory. This procedure is carried out only once, after which the program enters an iterative loop consisting of three steps:

1. The program acquires implant information from the visualization kernel and checks which cells to update and afterwards automatically chooses whether local or global update should be applied.

2. The unknowns of the cells whose stiffness matrices are updated in step 1 are solved by a nested dissection based p-FEM solver [120] which can efficiently handle local modifications.

3. In the post-processing 3D stress distributions are evaluated on predefined resulting points located inside the voxels. The code for post-processing has been parallelized with OpenMP to reach maximum efficiency. After the post-processing stage, the 3D stress dataset is transferred back to the visualization kernel which visualizes the data and completes one iteration.

The structure of this subsection is organized as follow: the OpenMP technique is introduced in Section 5.3.5.1, the update of cell stiffness matrices is addressed in detail in 5.3.5.2, the p-FEM solver which is well suited for the computational steering system is described in Section 5.3.5.3.

5.3.5.1 Introduction to OpenMP

OpenMP (Open Multi-Processing) is an application program interface for shared memory and distributed shared memory multiprocessors [121]. OpenMP takes a directive-based approach for supporting parallelism. It provides a set of pragmas, run-time routines, and environment variables that programmers can use to specify shared-memory parallelism in Fortran, C, and C++ programs. It is a widely accepted specification and is supported by venders like Sun, Intel, IBM and SGI.

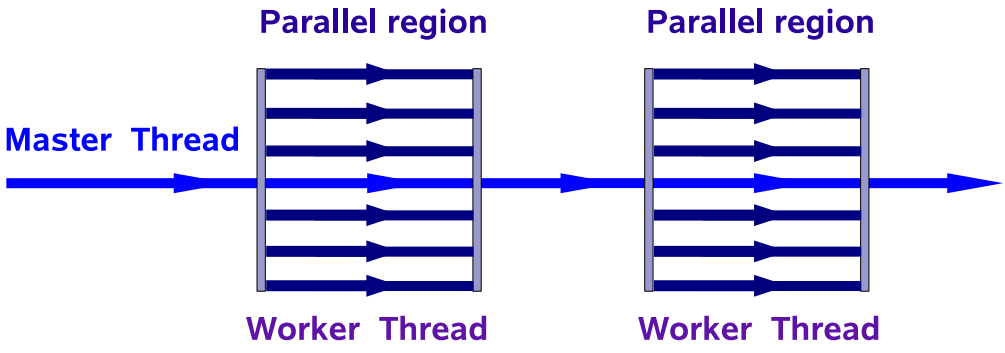


Figure 5.8: Fork and Join model

OpenMP is based on a Fork and Join model depicted in Figure 5.8. The program starts with a single process called the master thread, which runs sequentially until it reaches a parallel region. At this point additional threads, called the worker threads, are created and the master thread forks into the multiple worker threads. At the end of the parallel region the threads synchronize and join to become a single master thread again. Depending on usage, machine load and other factors, the threads are allocated to processors by the run-time environment. The number of threads can be assigned by the run-time environment based on environment variables ($OMP\_NUM\_THREADS$ in Linux system) or in code using functions.

OpenMP has the advantages of being very easy to employ on currently existing serial codes and allowing incremental parallelization. It takes parallel programming to the next level by creating and managing threads for the user. All one needs to do is to insert appropriate pragmas in the source program which is then compiled with an OpenMP supported compiler. OpenMP also has the advantages of being widely used, highly portable and ideally suited to multi-core architectures [122].

5.3.5.2 Update of stiffness matrices

The update of the stiffness matrices takes place in two cases. The first case is during the femur tissue removal process (refer to Section 5.3.2) when the two cutting planes are activated. As an approximation, voxels lying in between the two planes are set to void. For retaining the original computational domain, altered cells that contain only void voxels are not discarded. This updating process is carried out only once or of most a few times, therefore the stiffness matrices of the related cells are globally recomputed. The second case of update is during the implant steering process. In the visualization kernel the implant B-rep model is voxelized on the GPU each time its state is modified. To simulate the interaction between the implant and the femur, the following approach is implemented.

The computational domain and its discretization are kept fixed during the entire computational steering process, namely no remeshing is performed. The cell stiffness matrices remain unchanged when the implant is navigated outside the computational domain. Upon entrance of the implant model, the overlapping voxels which originally belong to the computational domain (either void or bone) are replaced by the implant voxels. More technically, the material properties of these voxels are assigned with the property of the implant. To reduce the computational complexity, implant voxels lying outside the computational domain are not considered. The stiffness matrices of the cells whose voxel members are modified by the implant model are marked to be recomputed. Three approaches are available for the recomputation:

1. A similar approach as in the initialization step is carried out. The stiffness matrices of all the marked cells can be computed globally with the BLAS subroutine.

2. Scalar-matrix multiplications are carried out locally for each voxel to compute the stiffness matrices of the overlapping voxels in all marked cells.

3. The drawback of the first approach is that performing a completely new computation on cells containing only a few overlapping voxels does not pay off. The second approach is a good

supplement to the first one; however, it lacks efficiency for cells containing a large number of overlapping voxels. Therefore, the third approach which combines the first and the second one is adopted. This approach is able to firstly determine cell-wise whether to update the cell stiffness matrix globally or locally, then to call the corresponding subroutine for computation. A threshold criterion is needed to judge whether global or local updating subroutine should be called. This criterion depends fully on the number of overlapping voxels in one cell (denoted as n_{ov}): if n_{ov} is larger than the threshold, the cell stiffness matrices are updated globally; otherwise updated locally. Computations with both subroutines have been carried out to quantify this threshold value empirically. In terms of the speedup graphs depicted in Figure 4.10 in Section 4.4.6.2, the speedup factor of using the BLAS routine depends on the number of cells and the polynomial degree. As a reasonable approximation, the influence of polynomial degree can be neglected since the graphs of $p = 1, ..., 6$ have the similar distribution. In the computational steering application it is reasonable to confine the total number of cell to below 500 due to the limitation of the computing power available. Accordingly the number of marked cells is also below 500. In terms of Figure 4.10, the speedup factor can be regarded as approximately constant within this range. As a result, the threshold value is independent of both p and the number of marked cells. With these considerations, the threshold value is empirically evaluated on 100 cells with shape functions of polynomial degree increasing from one to six in each cell.

In each evaluation the number of overlapping voxels of each cell (n_{ov}) increases uniformly from 1 to 1000. The computational time of both local and global update was measured on an Intel Xeon W5590, 3.33GHZ machine with 1 core activated. The threshold value is determined by finding out the value of n_{ov} with respect to which the global and local updating time is approximately the same. The threshold values with respect to different polynomial degrees are tabulated in Table 5.1.

Polynomial degree	1	2	3	4	5	6
Threshold	120	63	56	46	40	40

Table 5.1: Threshold values with respect to different polynomial degree p for total number of cells ≤ 500

Upon fixation of the threshold value, the marked cells can be categorized into two groups: one group is updated globally and the other group is updated locally. The number of globally updated cells is denoted as n_c^G and the number of locally updated cells is denoted as n_c^L. The subroutine for global update adopts the idea of transforming scalar-matrix multiplications into matrix-matrix multiplications which can be efficiently computed using the BLAS routine. In the subroutine for local update, stiffness matrices of the overlapping voxels in n_c^L cells are updated by performing scalar-matrix multiplications which are parallelized with OpenMP.

With this approach the optimum stiffness matrix updating time can be achieved. Further improvement on the performance of the computational steering system is shifted to assembly

and solver.

5.3.5.3 A fast direct solver based on the nested dissection algorithm

5.3.5.3.1 The Pardiso solver

The package PARDISO is a high-performance, robust, memory-efficient and easy to use library for solving large sparse symmetric and nonsymmetric linear systems of equations on shared-memory multiprocessors [88, 89]. This solver employs the *Metis nested dissection* reordering technique and shows advantages in stability and efficiency in solving sparse matrices over many other direct solvers. As described in Section 4.5, the Pardiso solver is utilized as the main solver for most FCM applications. It exhibits superior performance for solving linear systems of equations that are badly conditioned.

As the default setting for Pardiso, the input matrices must be stored in a compressed sparse row (CSR) format. Using this format, operating on the zero entries can be avoided and the unknowns of a sparse system of equations can be numbered in a way so that the total amount of storage required for the direct solution is reduced. However, the conversion of input matrix to this format is recursive and is required in every computational step. Moreover, the conversion process is unparallelizable and can not be performed locally, namely each time when the stiffness matrix of one or more cells is modified, the global matrix has to be completely reconverted. Impact of this drawback is not pronounced when the simulation is performed only once. However, in a computational steering environment, carrying out matrix conversion in each updating iteration reduces the efficiency to a notable extend. A more suitable solver is required for the computational steering system.

As an alternative, a nested dissection based p-FEM solver [120] is chosen and employed in the simulation kernel to handle the computational steering computations more efficiently.

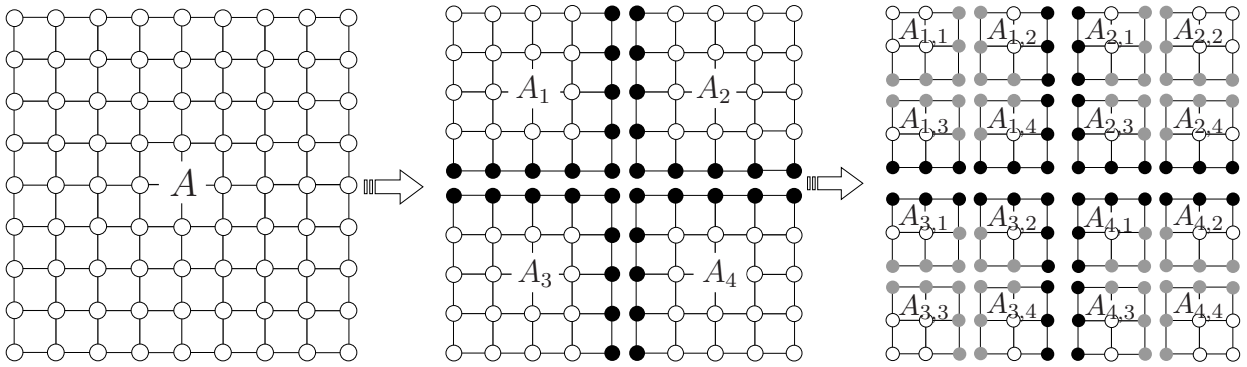
5.3.5.3.2 The nested dissection approach

Applying the nested dissection method as a direct solver was firstly introduced by George [123] for the solution of finite element problems. The basic idea of this method is to recursively subdivide the computational domain and eliminate local unknowns in a bottom-up step before in the final step a top-down solution process is performed. The algorithm can be split into three parts, recursive substructuring, static condensation, and solution [124].

1: Recursive substructuring

The idea is depicted on a mesh in Figure 5.9(a), which consists of nodal elements (hollow circles) interconnected by lines.

In the first step the initial computational domain A is partitioned into four subdomains (denoted as A_1, A_2, A_3, A_4) by a *separator*, which is a set of nodes marked in black in Figure 5.9(b). Next, each subdomain is further partitioned into four new subdomains (denoted as $A_{i,1}$, $A_{i,2}$, $A_{i,3}$, $A_{i,4}$, i= 1, 2, 3, 4) by a new *separator* marked in grey, see Figure 5.9(c). Following this strategy, the domain partitioning process is performed recursively until only



(a) Original 9 × 9 grids (b) First domain partitioning (c) Recursive domain partitioning

Figure 5.9: Nested dissection subdivision

few unknowns are consisted in each subdomain. To clarify the domain hierarchy an octree structure is employed to illustrate the domain subdivision, see Figure 5.10. The original domain is of the highest level (here Level 2), after each partitioning the newly generated four domains are of one level lower. The undividable subdomains are of the lowest level, level 0. The separator of each subdomain is set to have the same level number.

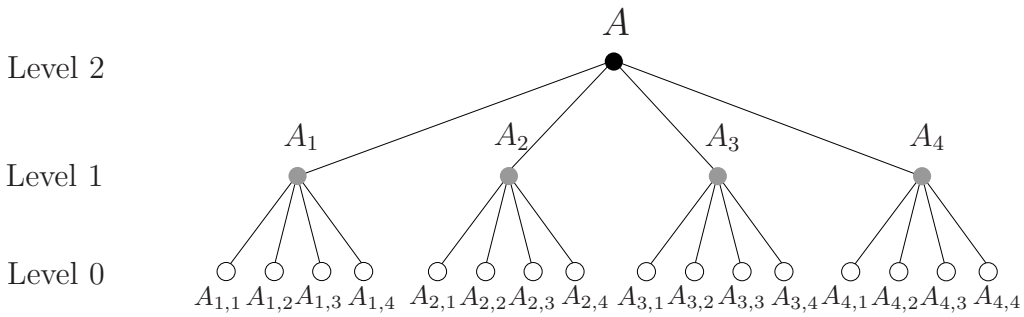


Figure 5.10: An octree structure

On each subdomain, the unknowns are categorized into two groups, inner unknowns and outer unknowns. The outer unknowns of subdomain level i are the unknowns on the ith level separator while the inner unknowns are the ones on the $i-1$th ($i \geq 2$) level separator. On the lowest level, the rest unknowns represented by the hollow circles in Figure 5.9(c) are called internal unknowns, whose coupling with the inner and outer unknowns can be eliminated by static condensation.

Referring to Figure 5.9(b) and 5.9(c), the separator that has a "+" shape is firstly used by George for domain partitioning. As an alternative, vertical and perpendicular separators with shapes like "−" and "|" can also be used to produce linear systems of equations with a slightly smaller number of unknowns [124].

2: Static condensation

This step computes the local system of equations of each subdomain. On the lowest level the system of equations is taken directly from the discretization. On higher levels the system is computed from the four subdomains. By reordering the unknowns according to inner (I) and outer (O) degrees of freedom, the system $Ku = d$ can be reassembled as:

$$\begin{bmatrix} K_{II} & K_{IO} \\ K_{OI} & K_{OO} \end{bmatrix} \begin{bmatrix} u_I \\ u_O \end{bmatrix} = \begin{bmatrix} d_I \\ d_O \end{bmatrix} \tag{5.1}$$

which can be written as

$$K_{II}\, u_I \;+\; K_{IO}\, u_O = d_I \tag{5.2}$$
$$K_{OI} u_I \;+\; K_{OO} u_O = d_O \tag{5.3}$$

The inner unknowns u_I from Equation (5.3) can be eliminated by solving Equation (5.2) for

$$u_I = K_{II}^{-1}(d_I - K_{IO}\, u_O) \tag{5.4}$$

which is substituted into Equation (5.3) to get

$$\tilde{K}_{OO} u_O = \tilde{d}_O \tag{5.5}$$

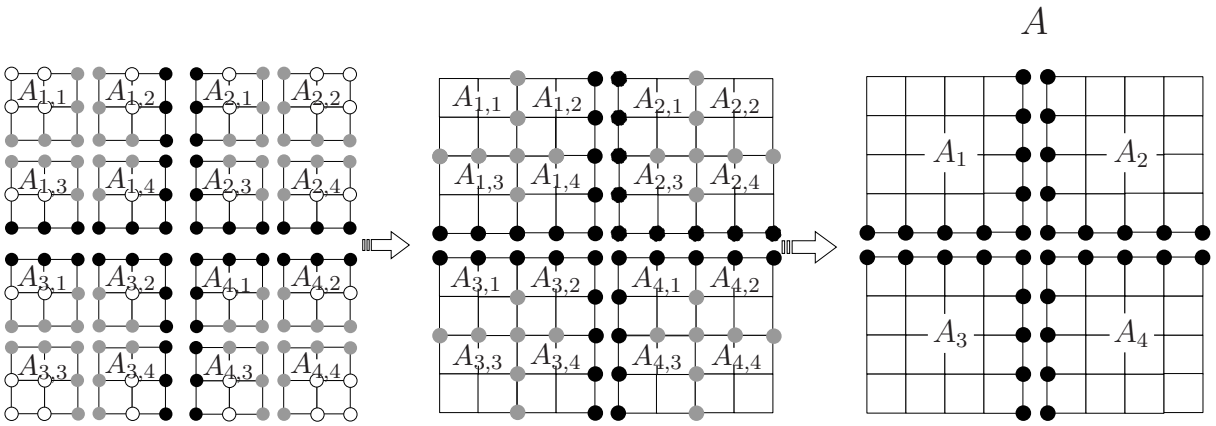
where $\tilde{K}_{OO} = (K_{OO} - K_{OI} K_{II}^{-1} K_{IO})$ and $\tilde{d}_O = d_O - K_{OI} K_{II}^{-1} d_I$.

The new system depends only on u_O. As a result, the cost of solving the complete local system of equations is reduced to solving part of the system only. The most expensive part of Equation (5.5) is to compute $\tilde{K}_{OO}$, the SCHUR complement of matrix K_{II}. There are several existing methods to compute the SCHUR complement, out of which *Gaussian elimination* is selected.

The static condensation procedure can be described using the octree hierarchy depicted in Figure 5.10. The static condensation is performed recursively in a bottom up step, starting from the leaf nodes of the tree. On each node, the inner (internal) unknowns are eliminated and the SCHUR complement matrix is computed and passed to the father node where a new system of equations is assembled with all the other SCHUR complement matrices of the son nodes. In Figure 5.11, the condensation procedure on the partitioned subdomains in Figure 5.9(c) is illustrated to provide a geometric description.

3: Solution

The solution process is started from solving the condensed system of equations on the highest level. Afterwards, the solution to the original system (Equation (5.1)) can be computed in a top-down process, where the values of the unknowns on the separators are computed recursively from the local systems of equations on each subdomain.



(a) Partitioned domain (b) Eliminating internal unknowns on the 0th level subdomains (c) Eliminating inner unknowns of the higher level subdomains

Figure 5.11: Nested dissection condensation

5.3.5.3.3 A nested dissection based p-FEM solver

Taking advantage of the hierarchical organization of the computational domain, the nested dissection scheme shows high efficiency in solving sparse systems of equations. Either octree or binary tree hierarchy can be used to discretize the geometry of the domain of a regular shape. However, it is difficult to model arbitrary shaped domains with a regular geometric discretization efficiently. To overcome this drawback, modifications of this method to solving problems with unstructured meshes have been proposed in several papers [125, 126]. One good strategy is to hierarchically discretize the finite elements instead of domain's geometry. Based on this strategy, a nested dissection approach for p-version FEM was developed by NIGGL and MUNDANI [120]. The main difference of this method from the general nested dissection approach lies in the domain substructuring step, which is split into two parts: finite element hierarchy creation and setup of degrees of freedoms.

1. Finite element hierarchy creation
After a finite element mesh is created, the elements are organized hierarchically using a spatial identifier − the barycenter. An octree is used to recursively subdivide the domain as long as more than one element center point is contained in an octree cell. A 2D sample of a finite element mesh and its hierarchical representation are depicted in Figure 5.12.

2. Setup of degrees of freedom
In contrast to the h-version, in the p-version finite element method the degrees of freedom are not only associated to nodes, but also to edges, faces and internal modes. Therefore, the DOFs which are located on the nodes, edges and faces shared by several elements must be assigned separately to the octree. The DOF setup procedure follows the rule that each DOF must be located at the lowest common father (LCF) of the element it belongs to. To find out the LCF of some elements, the so called Morton index is employed to identify the corresponding nodes in an octree. A numbering rule is applied to all the father nodes that its eight sons are numbered from 0 to 7 in a specific order. By accumulating all numbers on a way down from

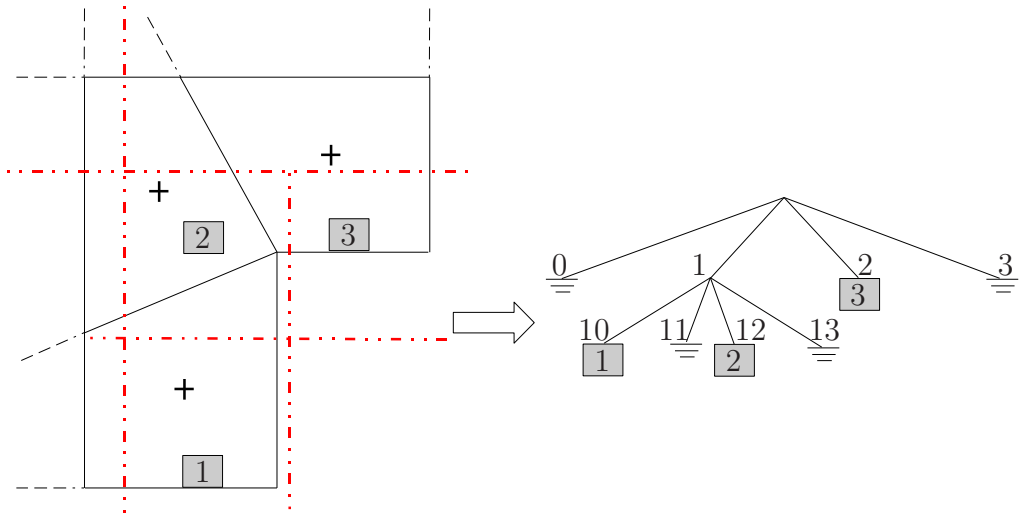


Figure 5.12: A 2D FE discretization and its corresponding octree structure. The numbering scheme of the octree is defined as: 0 − SW, 1 − SE, 2 − NW, 3 − NE.

the root to one node, the index of this node can thereby be obtained. A 2D FE discretization and its corresponding octree structure are depicted in Figure 5.12. After the Morton indices of all nodes are set up, the position of a DOF in the tree can be located by comparing the indices of the elements it belongs to. Figure 5.13 shows the DOF assignment of the 2D FE sample.

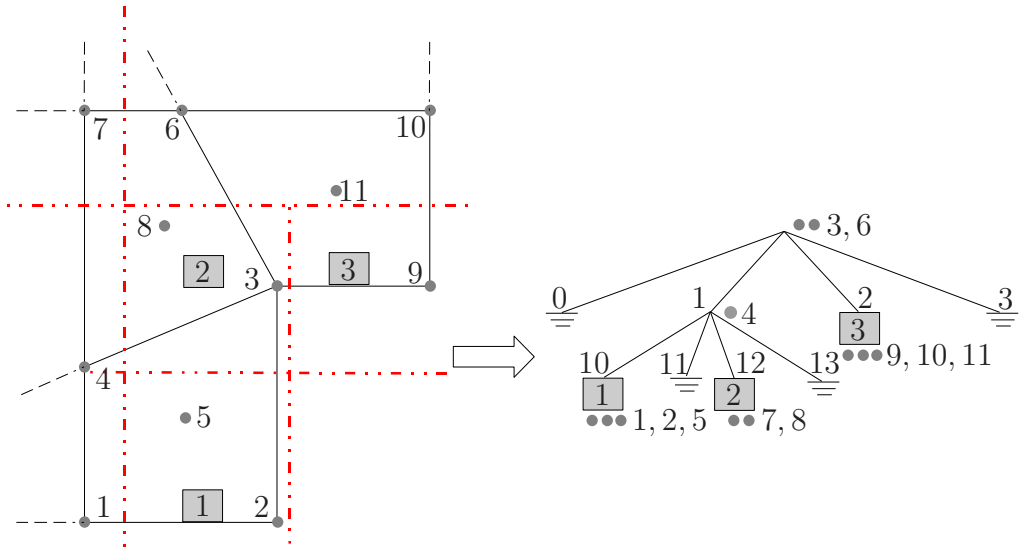


Figure 5.13: DOFs setup in a tree

The Morton indices of elements 1 and 2 are 10 and 12 respectively. A LCF of index 1 can be obtained by comparing the two Morton indices. The LCF index indicates the position where the common DOF shared by the 1st and 2nd elements − the DOF of number 4 is stored in the tree.

After all the DOFs are assigned, the element stiffness matrices and load vectors are computed and stored at the position of the particular element in the tree. This completes the step of domain substructuring. The next two steps, the static condensation and the solution are identical to the standard nested dissection approach and are applied to solve the system of equations.

The biggest advantage of this nested dissection method is its efficiency in performing local modifications. Instead of completely reassembling the system of equations, only parts of the tree on which local modifications take place have to be reassembled. This characteristics is highly suited for the computational steering environment in which only several cells undergo modification during the steering process while the rest remain unchanged.

5.3.6 Computational steering system construction with Internet sockets

The last key ingredient in the computational steering system is to couple the simulation kernel with the visualization kernel through an Internet socket connection. Description of the coupling details entails some basic knowledge of the Internet protocol and the Internet socket.

5.3.6.1 Internet Protocol

The Internet Protocol (IP) is the protocol by which data is transferred between computers on the Internet. The task of the Internet Protocol is to deliver distinguished protocol datagrams[2] from the source host to the destination host solely based on their addresses. For this purpose the Internet Protocol defines addressing methods and structures for datagram encapsulation [128]. The IP address is defined as a numerical label assigned to a computer or device in a network that uses the Internet Protocol to communicate between its nodes. With a given IP address computers can be specified via a network based on the fact that at a given time each host on the Internet has a unique IP address.

5.3.6.2 Internet socket

The Internet socket is a mechanism for exchanging data between processes, which can be either on the same or different machines connected via a network. Bidirectional data transfer is enabled by the establishment of a socket connection, until one of the endpoints terminates the connection [129].

An Internet socket consists of a local and a remote socket address. A socket address is a combination of an IP address and a port number which is utilized to distinguish multiple applications from each other. Sockets are typically used in conjunction with the Internet protocols − Transmission Control Protocol (TCP), Internet Protocol (IP) and User Datagram Protocol (UDP). TCP is a connection-oriented protocol that is responsible for reliable communication between two end processes. UDP is primarily used for broadcasting messages over a network.

[2] A datagram is a collection of data that is sent as a single message [127].

Sockets are usually implemented by using application programming interface (API) libraries such as Berkeley sockets (used on UNIX system) and Windows Sockets (used on Windows system). Socket interfaces can be divided into three types, datagram sockets which use UDP, stream sockets which use TCP, and raw sockets which use IP. The socket APIs are relatively small and simple. Many of the functions are similar to those used in file input/output routines such as read(), write(), and close(). The actual function calls to use depend on the programming language and the socket library chosen. Development of application programs that use the API is called socket programming or network programming [130, 131]. The TCP socket programming interface typically operates in a client-server model, the client sends out requests to the server, and the server does some processing with the requests received and returns a reply back to the client. A server-client model for a TCP socket connection is illustrated in Figure 5.14.

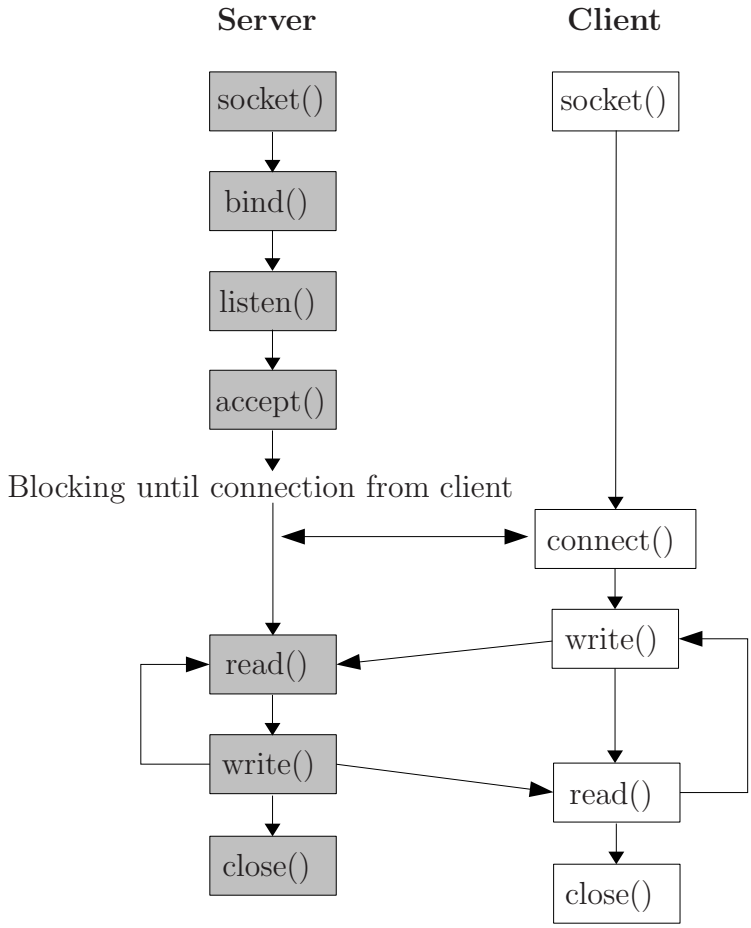


Figure 5.14: Socket system call

The system calls *socket*() and *bind*() first create a socket on the server which then waits for an incoming connection from the client using *listen*() and *accept*(). The client creates a socket by calling *socket*(), from which a request is sent to the server using *connect*() to establish a connection. After the establishment of connection, data are exchanged between the server and client using *read*() and *write*() until the termination of one endpoint.

5.3.6.3 Coupling of simulation and visualization

Three approaches for the implementation of computational steering are introduced in Section 5.2. They are: program instrumentation, direct scientific computation and recasting scientific computation. Being an easy-to-implement approach, program instrumentation is selected to reuse the simulation code *AdhoC* which is modified to provide functions called by the user via commands transferred through socket network. The socket system implemented for the computational steering is based on the server-client model depicted in Figure 5.14. After the socket connection is established, the user can manipulate the steering process through the user interface, which converts a user's operations into commands (integer numbers) that are sent to the simulation kernel. A list of enumerated commands is predefined and stored in both codes to ensure a common "language" for the communication. According to the command, the corresponding functions on the simulation kernel are called and executed. Dataflows, either incoming model information or outgoing stress results, are transferred in binary mode. In the steering mode, some functions are called iteratively to keep the results updated.

To enhance the communication efficiency, a dedicated data reduction scheme minimizing the dataflow to be transferred has been implemented. During the steering process when the implant is moved into the femur, only the indices (1D position number) of the altered voxels are computed and transferred. These indices (I_{1D}) are computed according to the formula

$$I_{1D} = I_z \cdot B_x \cdot B_y + I_y \cdot B_x + I_x \,, \tag{5.6}$$

where I_x, I_y and I_z are 3D position indices and B_x, B_y and B_z are the numbers of voxels in the data block. The computation takes only a small fraction of the total computational time. Hence, its impact can be neglected. After receiving the data, the recipient can retrieve the real positions by inversing Equation (5.6).

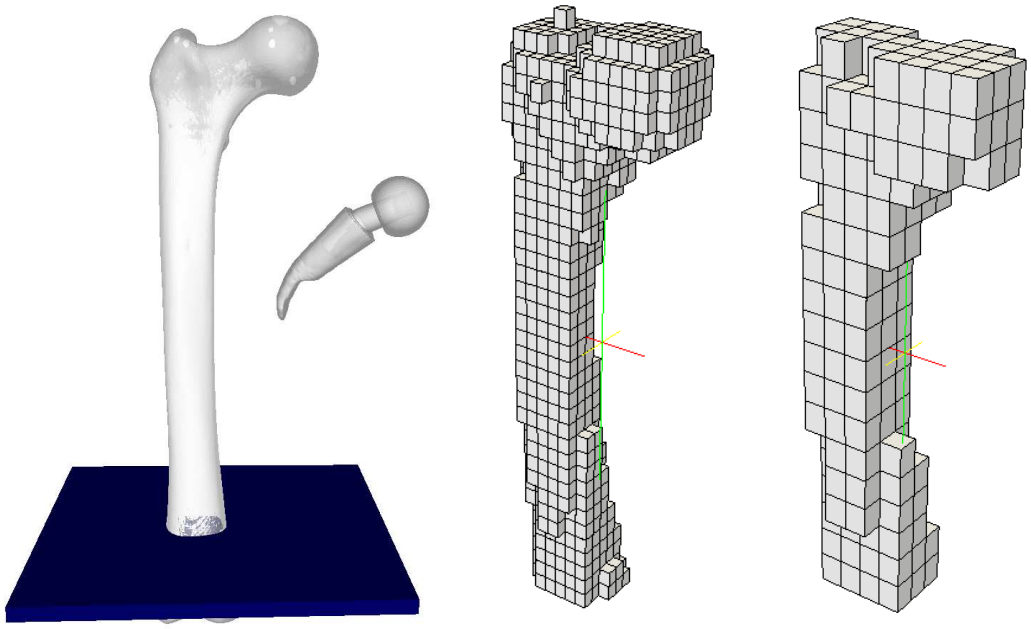
5.3.7 Surgical planning system demonstration

A femur model is obtained by CT scanning on a human femur specimen performed on a Siemens Sensation Cardiac 64 scanning machine with slice thickness of 1.0 mm and pixel size of 0.74 mm. The CT scans, having a resolution of $256 \times 256 \times 512$, are directly imported and the femur voxel model is segmented by thresholding the Hounsfield Unit values. Note that segmentation is not necessary for the FCM simulation, but only used to support the visualization of surface data of the femur. In addition to the femur model, the geometric model of a prosthesis is provided in STL format for generating implant voxel models. A material law relating the Hounsfield Unit value to the Young's modulus of each voxel as proposed in [132, 133] is employed. The Poisson's ratio is set to constant 0.3.

The material law

$$E = \begin{cases} 33900(8.2106 \cdot 10^{-4} HU + 0.057663)^{2.20} & HU \leq 320 \\ 10200(8.2106 \cdot 10^{-4} HU + 0.057663)^{2.01} & HU \geq 660 \\ 5307(8.2106 \cdot 10^{-4} HU + 0.057663) + 469 & 320 < HU < 660 \end{cases} \quad [MPa] \tag{5.7}$$

is provided by the user and is assigned in the code before the program starts. In this example, two resolution levels $1 \times 1 \times 1$ (new model has the same resolution as the original one) and $2 \times 2 \times 2$ (the resolution of the new model is half of the original one) are selected for the simulation.

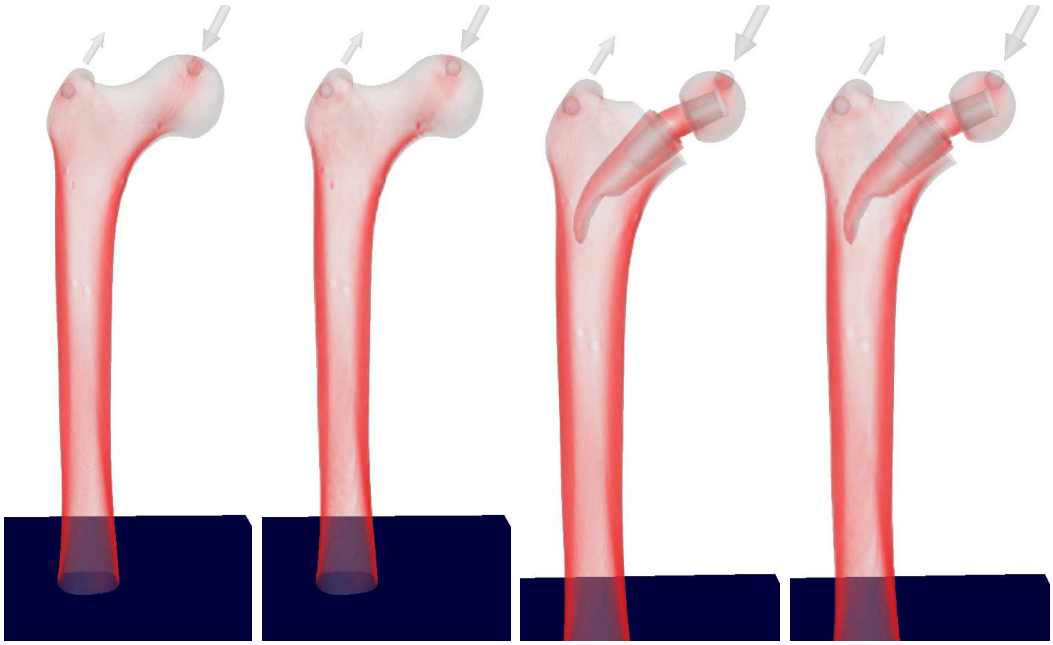


(a) Femur and implant model (b) Finite cell grids of the model with the original resolution (c) Finite cell grids of the model with half resolution

Figure 5.15: Femur model and its meshes

The femur model and the implant model are illustrated in Figure 5.15(a). After establishment of the socket connection, the segmented model and boundary conditions are sent to the simulation terminal, where computations are carried out. The number of voxels per cell is set to $10 \times 10 \times 10$ to minimize the total number of cells. From the voxel model finite cells are generated. Within each cell shape functions of polynomial degree four are used to attain high accuracy. For saving computational cost voxels below the fixation plane are not accounted for in the generation process of the grids. Cells that contain only void voxels are discarded leaving 1216 cells for the model at the original resolution level and 238 cells at the half resolution level, see Figure 5.15(b) and 5.15(c). Initially two loadings are activated: a compression load with a magnitude of 1500N located at the femur's head and a tension load with a magnitude of 1125.0N located at the greater trochanter. These two loads reflect the physiological loading imposed by the body weight and the muscles, respectively. In FCM the loads are applied on two small interior facets with the area of 4×4 voxel length horizontally located at the center of the loading spheres.

In the simulation both the preoperative physiological stress state and the altered stress state caused by the virtual implantation of a prosthesis model are computed and visualized using the aforementioned three visualization patterns.



(a) 1×1×1 resolution level (without implant) (b) 2 × 2 × 2 resolution level (without implant) (c) 1 × 1 × 1 resolution level (with implant) (d) 2 × 2 × 2 resolution level (with implant)

Figure 5.16: von Mises stress distributions visualized in red through ray casting

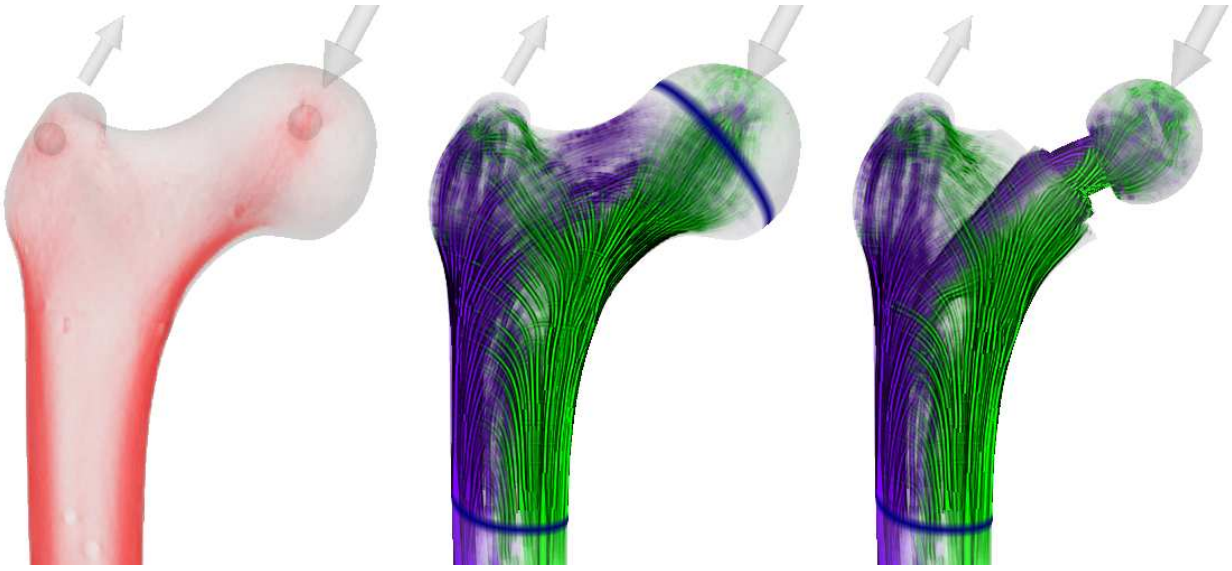
Figure 5.16 captures the screenshots from a running computation and displays the von Mises stress distributions of both models in red using volume rendering through ray casting [112]. The simulation results visualized in all figures show good accordance with physical intuition. Although theoretically the coarser resolution model produces results with a lower accuracy, apparent differences in von Mises stress indicated by the color intensity can not be observed due to limitations of this visualization pattern. The computational time for models at two resolution levels was measured and is tabulated in Table 5.2. Note that this time includes visualization, simulation and communication. The visualization time is of one or two magnitudes lower than the simulation time and is not accounted for in the statics.

The initialization time includes the time of CT data import and transferring, precomputed matrices setup, and initial stiffness matrices computation, etc. Unlike in many computational steering systems where a large portion of the time has to be spent in the initialization step, this system has a quick starting time. The steering time consists of the time of stiffness matrices update, solution, post-processing, and communication. For the finest model (1×1×1) one new computation (updating time) takes 8.75s which is, unfortunately, beyond an acceptable time range for a computational steering. Yet for the coarsened model (2×2×2) one new computation takes much less time − 0.75s allowing for a quasi real-time computational steering.

An enhanced pattern to the von Mises stress visualization is to visualize the principle stress using violet (tension) and green (compression) trajectories [113] as shown in Figure 5.17 and 5.18. Figure 5.17(b) depicts the femur interior stress lines at the physiological state under a body load. With the same data set, the virtual post-operative stress lines are plotted in 5.17(c). The implantation of a prosthesis considerably changes the stress distribution and in the new state

Model		**1×1×1**	**2×2×2**
DOF		61,368	13,698
Initialization time [s]	System setup	4.73	2.50
	Stiffness matrices computation	0.78	0.16
	Total	**5.51**	**2.66**
Updating time [s] (one iteration)	Stiffness matrices update	0.13	0.04
	Solution	6.94	0.50
	Post-processing	1.28	0.18
	Communication	0.22	0.03
	Total	**8.57**	**0.75**

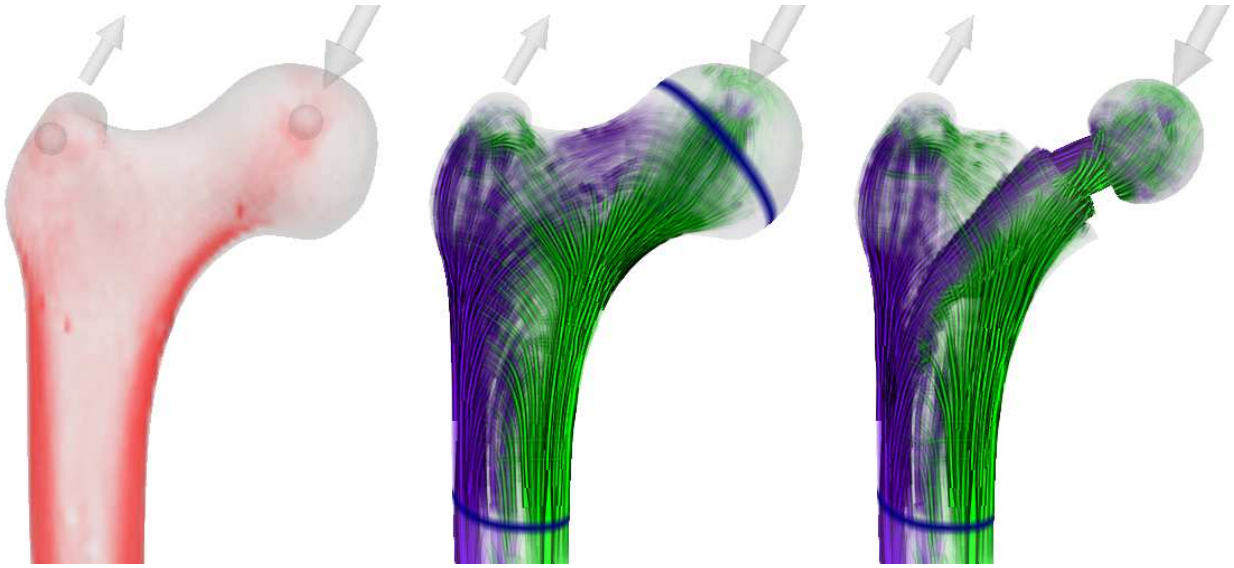
Table 5.2: Computational steering time measured on a Dell Precision T5500, 24GB Memory, Intel Xeon W5590 CPU, 3.33GHz, 8 cores



(a) von Mises stress distribution (b) principle stress lines (without implant) (c) principle stress lines (with implant)

Figure 5.17: Two visualization patterns on the $1 \times 1 \times 1$ level voxel model

the stress lines are mostly attracted by the prosthesis. This phenomenon coincides very well with the stress shielding effect mentioned in Section 5.1. To allow a more precise comparison, the third visualization pattern which exhibits only changes in stresses with yellow (decrease) and red (increase) colors is depicted in Figure 5.6(c). This novel way of visualization provides a more direct perception of changes in stress, with which decisions on the selection of an implant can be made relatively easily.



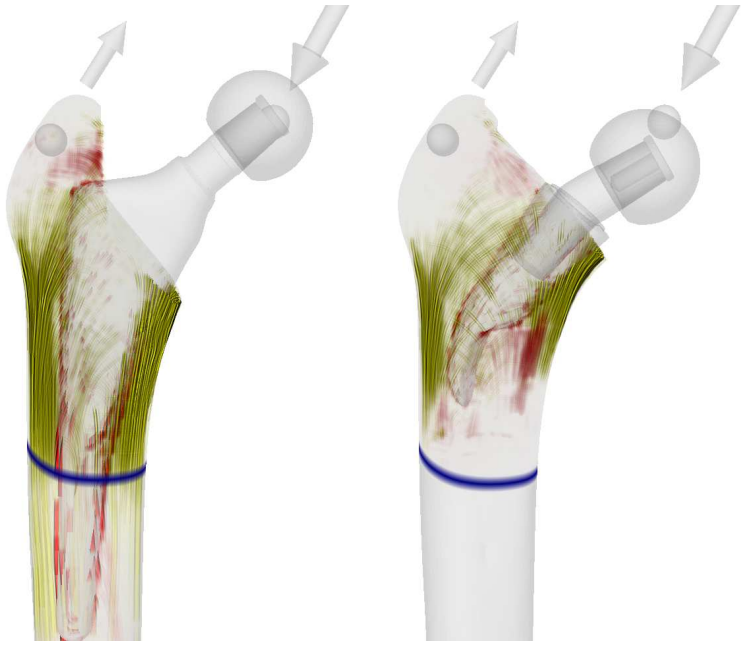
(a) von Mises stress distribution (b) principle stress lines (without implant) (c) principle stress lines (with implant)

Figure 5.18: Two visualization patterns on the $2 \times 2 \times 2$ level voxel model

In the following part of the example, an illustrative pre-operative prosthesis selection procedure by applying comparative stress visualizations is suggested. The goal is to show how the system can ease the selection of a suitable implant based on a patient-specific femur model. The influences of implant type, implant size and implant position are taken into account. In total two implant types, each having three sizes are tested in order to illustrate their impact on the post-operative stress distributions. The testing is separated into two parts: in the first part, two implant models each representing one type are tested and compared, after which one implant type is determined for the second part of testing. In the second part, implant models of the selected implant type with different sizes are tested with respect to different implantation positions. Finally by comparing the resulting comparative stresses, the best-fitting implant model and its positioning is determined.

In order to allow an interactive selection of implant, a suitable resolution level of the femur model must be preselected to ensure a smooth steering process before the program is started. This resolution level depends fully on the initial resolution of the femur model. Generally, a CT model obtained directly from standard CT images having a resolution of 256×256 can produce more than 600 cells, which are too many for a smooth steering. Hence, a lower resolution level - $2 \times 2 \times 2$ is chosen, resulting in around 200 cells for the current CT model. The error induced by the coarsened model is analyzed and quantified in a simpler biomechanical example presented in Section 5.3.8. In the current example, additional computations on the model with full resolution are performed in order to provide visual references for comparison. As a more general implementation, computations at full resolution can be used for fine-tuning.

1st Part: The implant type is determined according to the comparative stresses computed on two implant models of different types at a fixed implantation position. Figure 5.19 depicts the comparative stress visualizations of the normal components on the two models. One



(a) Implant model *G2* (b) Implant model em CUT

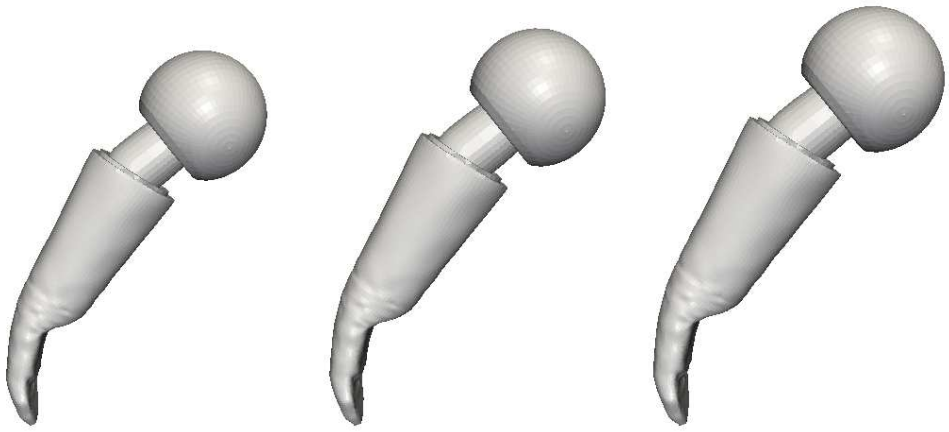
Figure 5.19: Comparative visualization of post-operative stress changes with respect to different implant models (both pictures are visualized based on the same color scale)

Figure 5.20: Two implant positions

can read from Figure 5.19(a) that the commonly used *G2* implant yields a strong increase in stress at the region of the greater trochanter and moreover a larger region of stress reduction in the femur neck and shaft. These changes indicate larger region of stress shielding and higher possibility of bone loss. In contrast to the *G2* implant, the new generation implant *CUT* has a much shorter stem which restricts the stress shielding zone to a smaller region, see Figure 5.19(b). As a result, instead of *G2* the implant type *CUT* is chosen in the first place.

2nd Part: Interactive testings of different implant models were performed at the $2 \times 2 \times 2$

resolution level. To ensure the correctness of a selection, recomputations of the comparative stress distributions at the $1 \times 1 \times 1$ resolution level were carried out to provide reference solutions. As an approximation to the real implantation context, instead of placing an implant in the interactive surgery at an arbitrary position, the central point of the implant head is fixed to the center of the femur head to allow only rotations of the stem. The determination of the positioning of the implant is thus simplified to only rotation. By studying the stress changes via rotating the implant stem, the two positions shown in Figure 5.20 are chosen for comparison.



(a) C_I (b) C_{II} (c) C_{III}

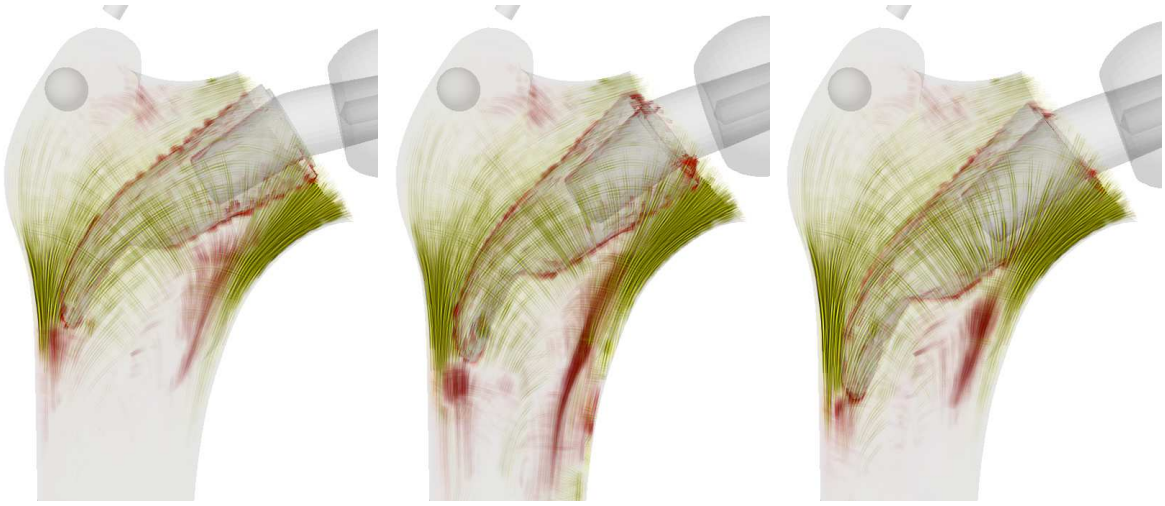
Figure 5.21: CUT Implant models

Figure 5.21 depicts three implant candidates with the same shape but increasing sizes. In the following they are referred to as C_I, C_{II} and C_{III}.

From the geometric perspective, model C_I is suited for the higher position while C_{II} and C_{III} are suited for the lower position. Interactive testings were performed to test the three cases at the $2 \times 2 \times 2$ level. Screenshots of the normal components of the comparative stress visualization for the three implant models were taken and are displayed in Figure 5.22. From these figures it can be deduced that for this patient-specific femur model, implantation with the prosthesis model C_I produces the least changes in normal stresses (indicated by smaller region of changes in stress and lower line intensity in both red and yellow colors), while C_{II} and C_{III} impose more stress shielding.

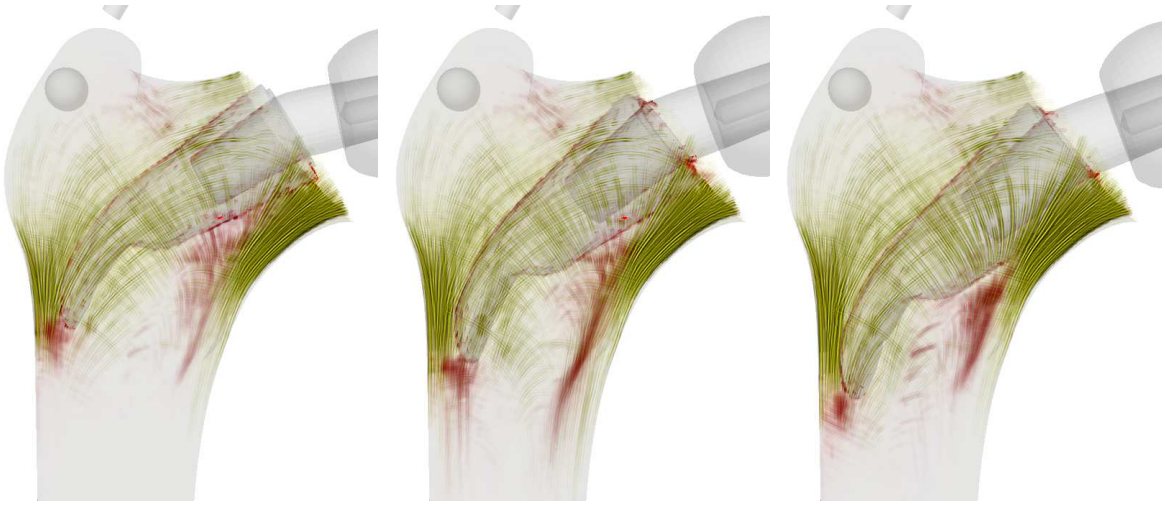
Hence it makes sense to select the model candidate C_I as the solution for this patient to minimize stress shielding. To prove our selection and to show more accurate results, computations at the $1 \times 1 \times 1$ resolution level were performed and the results are depicted in Figure 5.23. Differences between the two levels of computation can be observed. They are, however, not pronounced which indicates that the lower level of resolution chosen for the computational steering already provides sufficient accuracy to judge over the stability of an implant.

In addition to the normal components, the shear components of the comparative stress which affect the rotational stability of the post-operative integration of implant are also considered.



(a) Implant C_I (b) Implant C_{II} (c) Implant C_{III}

Figure 5.22: Normal components of the comparative stress visualization at the $2 \times 2 \times 2$ resolution level

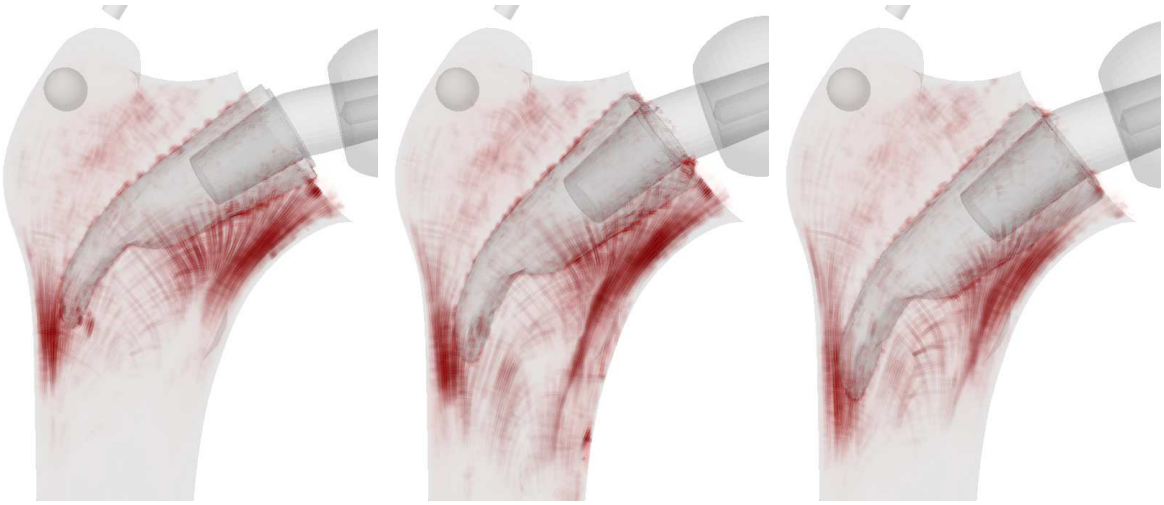


(a) Implant C_I (b) Implant C_{II} (c) Implant C_{III}

Figure 5.23: Normal components of the comparative stress visualization at the $1 \times 1 \times 1$ resolution level

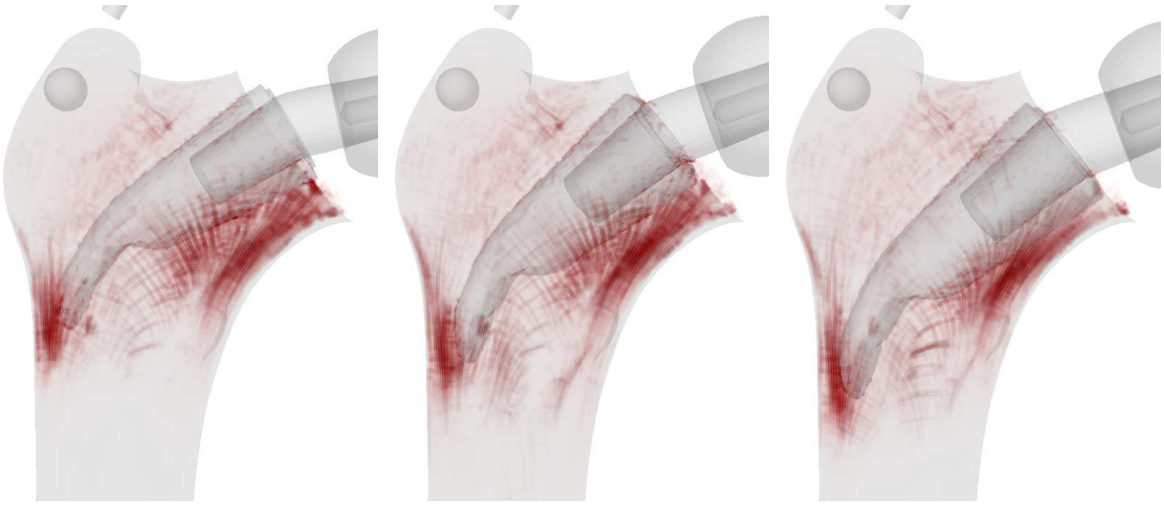
Comparative visualizations of the shear components are shown in Figure 5.24 and 5.25. According to the area and intensity of the red curves which indicate the range and magnitude of shear components, the largest implant C_{III} yields the best performance.

With the help of the results of the proceeding simulations, surgeons can now select either implant C_I for minimum stress shielding or implant C_{III} to achieve a higher implant rotational stability. The final decision depends on the surgeon's judgment in terms of the patient's need.



(a) Implant C_I (b) Implant C_{II} (c) Implant C_{III}

Figure 5.24: Shear components of the comparative stress visualization at the $2 \times 2 \times 2$ resolution level



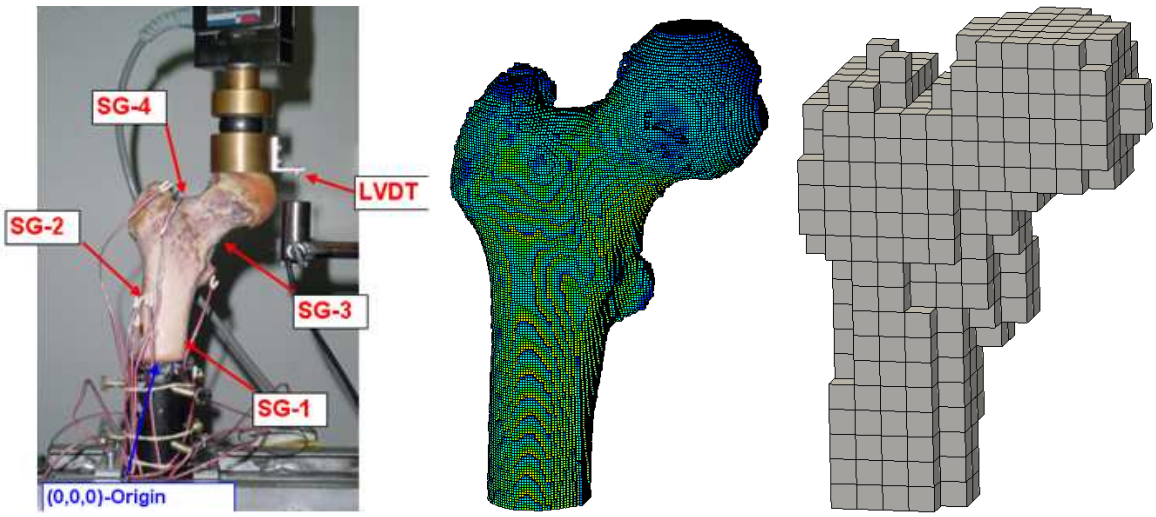
(a) Implant C_I (b) Implant C_{II} (c) Implant C_{III}

Figure 5.25: Shear components of the comparative stress visualization at the $1 \times 1 \times 1$ resolution level

5.3.8 Validation of the surgical planning system by a proximal femur experiment

This subsection presents the validation of the planning system by means of an example in which the results of the FCM computation are compared to in vitro experiments on the fresh-frozen proximal femur which were performed by YOSIBASH et al. [22].

The bone depicted in Figure 5.26(a) was completely fixed on the bottom and loaded by a load controlled machine with a constant value of 1000N on the femur top in the vertical direction. A Solartron DFg5 direct current linear variable displacement transducer (DC-LVDT) was positioned on a stand arm with its core connected to the loading machine to measure the femur



(a) (b) (c)

Figure 5.26: a: In vitro experiment, b: Femur voxel model, c: finite cell meshes

head vertical displacement. Four uniaxial strain gauges (SGs) were installed on the surface of the proximal femur at inferior and superior parts of the femur neck and on the medial and lateral femur shaft, see Figure 5.26(a). The strain values measured by the four SGs and the displacement value measured by the LVDT are used as reference solutions for the FCM comparison.

The FCM computations were conducted with the procedures detailed in Section 4.4. A femur voxel model which is extracted from a QCT scan taken by Yosibash et al. is depicted in Figure 5.26(b). The QCT scan has a resolution of $512 \times 512 \times 200$ with a spacing of $0.78125mm$ in the in-plane directions and $0.75mm$ in the longitudinal direction. As a reasonable approximation, the femur is assumed to be an inhomogeneous isotropic linear elastic solid. The voxel Hounsfield unit value as measured by a CT scan is converted to the bone equivalent density according to

$$\rho_{EQM[g/cm^2]} = 10^{-3} \left(0.6822\, HU - 5.48\right) \tag{5.8}$$

The bone density then is related to the Young's modulus by:

$$E = \begin{cases} 10200 \left(1.22\, \rho_{EQM} + 0.0523\right)^{2.01} & HU \leq 500 \\ 5307 \left(1.22\, \rho_{EQM} + 0.0523\right) + 469 & HU \geq 500 \quad [MPa] \end{cases} \tag{5.9}$$

A finite cell mesh was generated from the femur voxel model. Cells that contain only voids are discarded, the remaining 667 cells are shown in Figure 5.26(c). In the FCM analysis the femur bottom is fixed in three directions and a loading with a value of $1000N$ is applied on the top in order to simulate the real loading condition. The analysis is conducted by a uniform

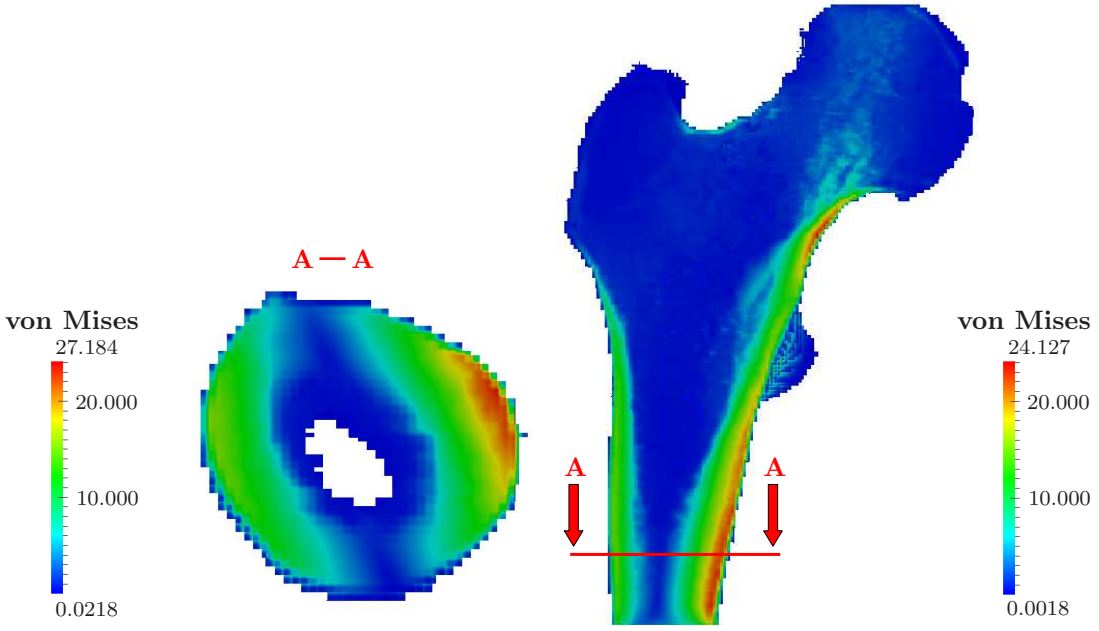


Figure 5.27: Cross-sectional views of the von Mises distribution in the proximal femur

p-extension with $p = 1, ..., 6$. The von Mises stresses are plotted in x-y and x-z cross-sectional views shown in Figure 5.27. The convergence curve of the strain energy is plotted in Figure 5.28.

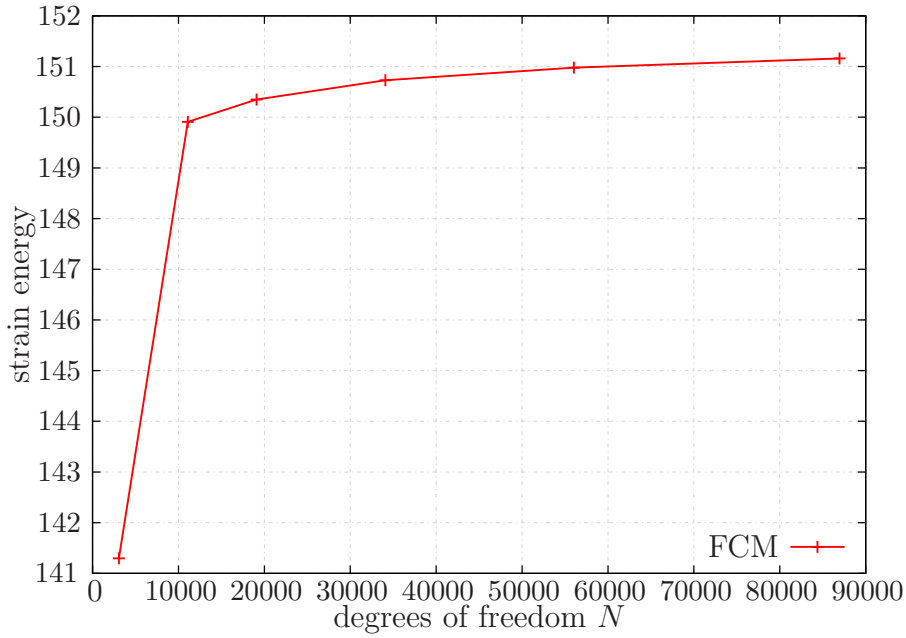


Figure 5.28: Convergence of strain energy

The principle strains computed by the FCM with $p = 4$ are listed together with the experimentally measured uniaxial strains in the upper part of Table 5.3. In the FCM computation, the vertical displacement on the top of the femur is approximated by a point-wise displacement evaluated on a point located in the center of the top surface. Two types of strain values are

presented: the "point-wise" value gives the principle strain evaluated at exactly the center point of the strain gauge, while the "surface averaged" value is computed by averaging the principle strains over the strain gauge surface due to the fact that SG readings are an averaged value over the length of the SGs.

		LVDT	**SG1**	**SG2**	**SG3**	**SG4**
Experiment [$\mu\varepsilon$]	**Surface**	-300	-1303	440	-1303	441
FCM results [$\mu\varepsilon$] ($1 \times 1 \times 1$)	**Point-wise**	-295	-1249	806	-1171	436
	Error^e[%]	1.67	4.14	83.18	10.13	0.91
	Surface	NA	-1246	765	-1154	440
	Error^e[%]	NA	4.37	73.86	11.43	0.22
FCM results [$\mu\varepsilon$] ($2 \times 2 \times 2$)	**Point-wise**	-291	-1287	786	-1174	479
	Error^e[%]	3.00	1.23	78.63	9.90	8.86
	Error^f[%]	1.36	3.04	2.48	0.26	9.86

Table 5.3: Comparison of the FCM strain results ($p = 4$) with experiment. Error^e denotes the relative error with respect to the experiment, while Error^f denotes the relative error with respect to the FCM results of the model at a higher resolution level

A good agreement between the FCM computation and the experiment is observed except at the second strain gauge. Concerning this point, the computation were verified by comparing with the FEM result of Yosibash who performed the same analysis with the standard p-FEM. The average strain value obtained by Yosibash on the second SG is 790 [$\mu\varepsilon$], which is in good accordance with the FCM result. Hence the large error on SG2 is attributed to a defaulting of the strain gauge during the experiment.

In [56] a finite element analysis of human bone was performed based on a coarsened voxel model derived from the voxel model with full resolution. Such a coarsened model has been used in Section 5.3.7 for the purpose of enabling the computational steering. To evaluate and quantify the accuracy of computation with coarser models, an additional computation was carried out on a model coarsened from the resolution of $1 \times 1 \times 1$ to $2 \times 2 \times 2$. The coarsening is accomplished by averaging the Young's modulus of eight neighboring voxels. The new model is then meshed with 124 cells. For comparison purpose only $p = 4$ is used in each cell. With the same boundary conditions and SG positions the FCM results of the second model are shown in the lower part of Table 5.3. The accuracy of the FCM computation is lower on the coarsened model than on the original one, but is still within an acceptable range.

Measured on an Intel Xeon W5590, 3.33GHz work station with 8 cores, the total computational time and one computational steering iteration time for the original model (computed with $p =$

4) is approximately 9 and 4 seconds respectively after parallelization, while for the coarsened model is 4 seconds and 0.5 seconds.

Chapter 6

Conclusions

The finite cell method with fast integration is presented in this thesis. This scheme shows a remarkable efficiency in computational time on three-dimensional CT-derived models with complex geometries and multi-material interfaces, like for instance biological hard tissues. Three numerical examples are given to demonstrate the accuracy of this method by comparison with either analytical or numerical reference solutions.

Development of the FCM with fast integration provides a new yet efficient way to handle biomechanical simulations which naturally possess complicated geometries. Fast simulations on these geometries, which are difficult to be performed using the traditional FEM are thus made possible. This method can be implemented in various applications, e.g., a fast and interactive tool for patient-specific surgery planning, an efficient homogenization tool for micro-CT structures, or an analysis tool for probabilistic analysis of human bone's mechanical behavior. Further applications of this method are not limited to biomechanical simulations only; any voxelizable physical model fulfilling the requirement (with either complex geometry or inhomogeneous material properties) can be efficiently simulated. However it is not recommended to implement this scheme for simple structures with regular geometries, which can be solved with little effort using standard FEM.

Powered by the FCM with fast integration, an interactive preoperative surgical planning system has been constructed and is presented in this dissertation. With the advanced simulation technique and the GPU-based advanced visualization techniques, a quasi real-time computational steering system is created and can be used for an interactive selection of prostheses. A planning example is given to demonstrate the efficiency of the system and to clarify the implant selection procedure. Being a new approach to visualize post-operative stress changes, the comparative stress visualization is demonstrated to be an effective means for selecting the type, size and positioning of a patient-specific implant.

The current planning system can provide the images of stress on the screen, according to which an implant is selected based on the judgment of a surgeon. Further improvement of this system can be to develop a deterministic selection program, which can determine a unique implant size and positioning automatically. Furthermore, as the development of computing power, enabling computational steering at the highest resolution level may be made possible in the near future.

Bibliography

[1] Z. Yosibash, R. Padan, L. Joscowicz, C. Milgrom, A CT-based high-order finite element analysis of the human proximal femur compared to in-vitro experiments, ASME Journal of Biomechanical Engineering 129 (2007) 297–309.

[2] J. Parvizian, A. Düster, E. Rank, Finite cell method – h- and p-extension for embedded domain problems in solid mechanics, Computational Mechanics 41 (2007) 121–133.

[3] A. Düster, J. Parvizian, Z. Yang, E. Rank, The Finite Cell Method for three-dimensional problems of solid mechanics, Computer Methods in Applied Mechanics and Engineering 197 (2008) 3768–3782.

[4] R. Padan, Towards a Reliable Mechanical Simulation of the Proximal Femur, Master's thesis, Ben-Gurion University of the Negev, Beer-Sheva (2006).

[5] C. Clemente, Gray's Anatomy of the Human Body, Lea & Febiger, 1985.

[6] V. C. Mow, R. Huiskes, Basic Orthopaedic Biomechanics and Mechano-Biology, Lippincott Williams & Wilkins; Third Edition, 2004.

[7] University of London, Bone Ossification & Growth, `https://courses.stu.qmul.ac.uk/smd/kb/microanatomy/bone/index.htm`.

[8] A. P. Spence, Basic Human Anatomy, Benjamin-Cummings Pub Co; Third Edition, 1990.

[9] K. U. Leuven, Biomechanics in Dentistry, `http://www.feppd.org/ICB-Dent/campus/biomechanics_in_dentistry/ldv_data/mech/basic_bone.htm`.

[10] G. F. Knoll, Radiation Detection and Measurement, John Wiley & Sons (4th Edition), 2010.

[11] G. T. Herman, Foundamentals of computerized tomography: Image reconstruction from projection, Springer, 2nd Edition, 2009.

[12] S. Henwood, Clinical CT: techniques and practice, Cambridge University Press, 1999.

[13] M. Prokop, Spiral and Multislice Computed Tomography of the Body, Thieme, 2002.

[14] R. Pelberg, W. Mazur, Cardiac CT Angiography Manual, Springer, 2007.

[15] D. Mukherjee, S. Rajagopalan, CT and MR Angiography of the Peripheral Circulation: Practical Approach with Clinical Protocols, CRC Press, 2007.

[16] I. Leichter, A. Bivas, A. Giveon, J. Y. Margulies, A. Weinreb, The relative significance of trabecular and cortical bone density as a diagnostic index for osteoporosis, Physics in Medicine and Biology 32(9) (1987) 1167–1174.

[17] H. N. Herkowitz, G. R. Bell, The lumbar spine, Lippincott Williams & Wilkins, 2004.

[18] A. Terrier, J. Miyagaki, H. Fujie, K. Hayashi, L. Rakotomanana, Delay of intracortical bone remodelling following a stress change: A theoretical and experimental study, Clinical Biomechanics 20 (2005) 9981006.

[19] E. Schileo, E. DallAra, F. Taddei, A. Malandrino, T. Schotkamp, M. Baleani, M. Viceconti, An accurate estimation of bone density improves the accuracy of subject-specific finite element models, Journal of Biomechanics 41 (2008) 2483–2491.

[20] R. E. Kusy, T. C. Peng, P. E. Hirsch, S. C. Garner, Interrelationships of Bone Ash and Whole Bone Properties in the Lactating and Parous Rat, Calcif Tissue Int 41 (1987) 337–341.

[21] W. T. Edwards, R. C. McBroom, W. C. Hayes, Variation of density in the vertebral body measured by quantitative computed tomography, Trans. Orthop. Res. Soc. 11 (1986) 205.

[22] Z. Yosibash, N. Trabelsi, C. Milgrom, Reliable simulations of the human proximal femur by high-order finite element analysis validated by experimental observations, Journal of Biomechanics 40 (2007) 3688–3699.

[23] A. H. Karantanas, J. A. Kalef-Ezra, D. C. Glaros, Quantitative computed tomography for bone mineral measurement: technical aspects, dosimetry, normal data and clinical applications, British Journal of Radiology 64 (1997) 298–304.

[24] M. Alter, A. Rogers, Science of Flexibility, Human Kinetics Publishers; 3rd Edition, 2004.

[25] T. M. Keaveny, X. E. Guo, E. F. Wachtel, T. A. McMahon, W. C. Hayes, Trabecular bone exhibits fully linear elastic behavior and yields at low strains, J Biomech 27(9) (1994) 1127–1136.

[26] A. Mortensen, Concise encyclopedia of composite materials, Elsevier Science; Second Edition, 2007.

[27] J. C. Rice, S. C. Cowin, J. A. Bowman, On the dependence of the elasticity and strength of cancellous bone on apparent density, J Biomech. 21 (1988) 155–168.

[28] S. C. Cowin, G. Yang, Averaging Anisotropic Elastic Constant Data, Journal of Elasticity 46 (1997) 151–180.

[29] J. C. Lotz, T. N. Gerhart, W. C. Hayes, Mechanical properties of trabecular bone from the proximal femur: a quantitative CT study, J Comput Assist Tomogr 14(1) (1990) 107–114.

[30] T. S. Keller, Predicting the compressive mechanical behavior of bone, Journal of Biomechanics 27(9) (1994) 1159–1168.

[31] J. Catanese, E. P. Iverson, R. K. Ng, T. Keaveny, Heterogeneity of the mechanical properties of demineralized bone, Journal of Biomechnics 32(12) (1999) 1365–1369.

[32] A. G. Au, A. B. Liggins, V. J. Raso, J. Carey, A. Amirfazli, Representation of bone heterogeneity in subject-specific finite element models for knee, Computer Methods and Programs in Biomedicine 99(2) (2010) 154–171.

[33] M. J. Ciarelli, S. A. Goldstein, J. L. Kuhn, D. D. Cody, M. B. Brown, Evaluation of orthogonal mechanical properties and density of human trabecular bone from the major metaphyseal regions with materials testing and computed tomography, Journal of Orthopaedic Research 9 (1991) 674682.

[34] J. Y. Rho, An ultrasonic method for measuring the elastic properties of human tibial cortical and cancellous bone, Ultrasonics 34(8) (1996) 777–783.

[35] S. M. Lang, D. D. Moyle, E. W. Berg, N. Detorie, A. T. Gilpin, N. J. Pappas, J. C. Reynolds, M. Tkacik, R. L. Waldron, Correlation of mechanical properties of vertebral trabecular bone with equivalent mineral density as measured by computed tomography, The Journal of Bone and Joint Surgery 70(10) (1988) 1531–1538.

[36] T. S. Kaneko, J. S. Bell, M. R. Pejcic, J. Tehranzadeh, J. H. Keyak, Mechanical properties, density and quantitative CT scan data of trabecular bone with and without metastases, Journal of Biomechanics 37(4) (2004) 523 – 530.

[37] P. Augat, T. Link, T. F. Lang, J. C. Lin, S. Majumdar, G. H. K., Anisotropy of the elastic modulus of trabecular bone specimens from different anatomical locations, Medical Engineering & Physics 20 (1998) 124–131.

[38] C. E. Hoffler, K. E. Moore, K. Kozloff, P. K. Zysset, S. A. Goldstein, Age, Gender, and Bone Lamellae Elastic Moduli, The Journal of Bone and Joint Surgery 18 (2000) 132–143.

[39] M. Hobatho, J. Rho, R. Ashman, Anatomical variation of human cancellous bone mechanical properties in vitro, Stud Health Technol Inform 40 (1997) 157–173.

[40] The Internet Encyclopedia of Science, Femur, `http://www.daviddarling.info/encyclopedia/F/femur.html`.

[41] The American Heritage Medical Dictionary, Houghton Mifflin Harcourt, 2008.

[42] M. Schünke, E. Schulte, U. Schumacher, PROMETHEUS Lernatlas der Anatomie. Allgemeine Anatomie und Bewegungssystem, Thieme, Stuttgart, 2005.

[43] L. Medical Internet Solutions, Anatomy of the Knee, `http://www.aclsolutions.com/anatomy.php`.

[44] J. C. Koch, The laws of bone architecture, Am. J. Anat. 21 (1917) 177–298.

[45] A. Maciel, Biomechanics of Hip Joint Capsule, Computer Graphics Lab, Institute of Computing and Multimedia Systems, School of Computer and Communication Sciences, Swiss Federal Institute of Technology (2002).

[46] J. Grunewald, Die Beanspruchung der Lagen Rohrenknochen des Menschen, Z Orthop Chir 39 (1920) 27–49.

[47] P. Marique, Etudes sur le femur, Libr. Sci. Bruxelles (1945) 1–180.

[48] E. C. Case, A possible explanation of fenestration in the primitive reptilian skull, with notes on the temporal region of the genus Dimetrodon, Contributions from the Museum of Geology, University of Michigan 2(1) (1984) 1–12.

[49] I. Hvid, Mechanical strength of trabecular bone at the knee, Dan Med Bull 35(4) (1988) 345–365.

[50] P. Lafortune, C. E. Aubin, H. Boulanger, I. Villemure, K. M. Bagnall, A. Moreau, Biomechanical simulations of the scoliotic deformation process in the pinealectomized chicken: a preliminary study, Scoliosis 2(16).

[51] R. Ruimerman, Modeling and remodeling in bone tissue, Ph.D. thesis, Technische Universiteit Eindhoven (2005).

[52] T. Ota, I. Yamamoto, R. Morita, Fracture simulation of the femoral bone using the finite-element method: How a fracture initiates and proceeds, Journal of Bone and Mineral Metabolism 17(2) (1998) 108–112.

[53] P. K. Tomaszewski, N. Verdonschot, S. K. Bulstra, G. J. Verkerke, A Comparative Finite-Element Analysis of Bone Failure and Load Transfer of Osseointegrated Prostheses Fixations, Annals of Biomedical engineering 38(7) (2010) 2418–2427.

[54] J. Cegonino, J. M. García Aznar, M. Doblaré, D. Palanca, B. Seral, F. Seral, A Comparative Analysis of Different Treatments for Distal Femur Fractures using the Finite Element Method, Computer Methods in Biomechanics and Biomedical Engineering 7(5) (2004) 245–256.

[55] M. Viceconti, L. Bellingeri, L. Cristofolini, A. Toni, A comparative study on different methods of automatic mesh generation of human femurs, Medical Engineering & Physics 20 (1998) 1–10.

[56] J. H. Keyak, J. M. Meagher, H. B. Skinner, C. D. Mote, Automated three-dimensional finite element modelling of bone: a new method, Journal of Biomedical Engineering 12 (1990) 389–397.

[57] B. v. Rietbergen, H. Weinans, R. Huiskes, A. Odgaard, A new method to determine trabecular bone elastic properties and loading using micromechanical finite-element models, Journal of Biomechanics 28(1) (1995) 69–81.

[58] D. P. Fyhrie, M. S. Hamid, R. F. Kuo, S. M. Lang, Direct three dimensional finite element analysis of human vertebral cancellous bone, in: Transactions of the 38th Annual Meeting of the Orthopaedic Research Society, ORS, 1992, p. 551.

[59] O. Zienkiewicz, R. Taylor, The Finite Element Method – The Basis, 5th Edition, Vol. 1, Butterworth-Heinemann, 2000.

[60] K. J. Bathe, Finite element procedures, Prentice Hall, 1996.

[61] T. J. R. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Dover Publications, 2000.

[62] B. A. Szabó, I. Babuška, Finite Element Analysis, John Wiley & Sons, 1991.

[63] P. D. Lax, A. N. Milgram, "parabolic equations". Contributions to the theory of partial differential equations., Annals of Mathematics Studies, Princeton University Press 33 (1954) 167–190.

[64] N. S. Ottosen, M. Ristinmaa, The Mechanics of Constitutive Modeling, Elsevier Science, 2005.

[65] I. Babuška, T. Strouboulis, The finite element method and its reliability, Oxford University Press, 2001.

[66] J. Stewart, Calculus: Concepts and Contexts, Brooks Cole, 2009.

[67] D. W. Pepper, J. C. Heinrich, The finite element method: basic concepts and applications, Taylor & Francis, 1992.

[68] H. Bungartz, M. Schäfer, Fluid-structure interaction, Springer, 2006.

[69] A. Düster, High order finite elements for three-dimensional, thin-walled nonlinear continua, Ph.D. thesis, Lehrstuhl für Bauinformatik, Fakultät für Bauingenieur- und Vermessungswesen, Technische Universität München (2001).

[70] A. Düster, H. Bröker, E. Rank, The p-version of the finite element method for three-dimensional curved thin walled structures, International Journal for Numerical Methods in Engineering 52 (2001) 673–703.

[71] P. Neittaanmäki, D. Tiba, An embedding of domains approach in free boundary problems and optimal design, SIAM Journal on Control and Optimization 33 (5) (1995) 1587–1602.

[72] B. A. Szabó, A. Düster, E. Rank, The p-version of the Finite Element Method, in: E. Stein, R. de Borst, T. J. R. Hughes (Eds.), Encyclopedia of Computational Mechanics, Vol. 1, John Wiley & Sons, 2004, Ch. 5, pp. 119–139.

[73] M. Ruess, D. Tal, N. Trabelsi, Z. Yosibash, E. Rank, The finite cell method for bone simulations: verification and validation, submitted to Biomechanics and Modeling in Mechanobiology.

[74] D. Schillinger, A. Düster, E. Rank, The hp-d adaptive finite cell method for geometrically nonlinear problems of solid mechanics, submitted to International Journal for Numerical Methods in Engineering.

[75] L. Zhang, T. Cui, H. Liu, A set of symmetric quadrature rules on triangles and tetrahedra, Journal of Computational Mathematics 27 [1] (2009) 89–96.

[76] J. Savage, A. Peterson, Quadrature rules for numerical integration over triangles and tetrahedra, IEEE Antennas and Propagation Magazine 38 [3] (1996) 100–102.

[77] G. R. Cowper, Gaussian quadrature formulas for triangles, Internal Journal for Numerical Methods in Engineering 7 [3] (1973) 405–408.

[78] S. Osher, R. Fedkiw, Level-Set Methods and Dynamic Implicit Surfaces, Springer, 2003.

[79] J. A. Sethian, Level-Set Methods and Fast Marching Methods, Cambridge University Press, 1999.

[80] F. W. Liou, Rapid prototyping and engineering applications, CRC Press, 2007.

[81] P. Wenisch, C. van Treeck, A. Borrmann, E. Rank, O. Wenisch, Computational steering on distributed systems: indoor comfort simulations as a case study of inter-active cfd on supercomputers, International Journal of Parallel, Emergent and Distributed Systems 22 (4) (2007) 275–291.

[82] Le Telephone, `http://www.3dvia.com/models/F86CE0EEC0D2E4F6/le-telephone`.

[83] B. v. Rietbergen, Computational Strategies for Iterative Solutions of Large FEM Applications Employing Voxel Data, International Journal for Numerical Methods in Engineering 39 (1996) 2743–2764.

[84] A. Alberich-Bayarri, D. Moratal, L. Marti-Bonmat, M. Salmeron-Sanchez, A. Valles-Lluch, L. Nieto-Charques, J. J. Rieta, Volume Mesh Generation and Finite Element Analysis of Trabecular Bone Magnetic Resonance Images, in: Conf Proc IEEE Eng Med Biol Soc, 2007, pp. 1603–1606.

[85] J. J. Dongarra, J. Du Croz, I. S. Duff, S. Hammarling, Algorithm 679: A set of Level 3 Basic Linear Algebra Subprograms, ACM Trans. Math. Soft. 16 (1990) 18–28.

[86] BLAS (Basic Linear Algebra Subprograms), `http://www.netlib.org/blas/index.html`.

[87] U. Heißerer, AdhoC^4 – Technical Guide, Lehrstuhl für Bauinformatik, Technische Universität München (2007).

[88] O. Schenk, K. Gärtner, Solving Unsymmetric Sparse Systems of Linear Equations with PARDISO, Journal of Future Generation Computer Systems 20(3) (2004) 475–487.

[89] O. Schenk, K. Gärtner, On fast factorization pivoting methods for symmetric indefinite systems, Elec. Trans. Numer. Anal. 23 (2006) 158–179.

[90] A. Henderson, ParaView Guide, A Parallel Visualization Application, Kitware Inc (2007).

[91] E. Perilli, F. Baruffaldi, M. Visentin, B. Bordini, F. Traina, A. Cappello, M. Viceconti, MicroCT examination of human bone specimens: effects of polymethylmethacrylate embedding on structural parameters, Journal of Microscopy 225 (2007) 192–200.

[92] F. Baruffaldi, E. Perilli, Human bone biopsy, 3D model Bn_326_3D.stl, From: The BEL Repository, http://www.tecno.ior.it/VRLAB/.

[93] J. Schöberl, NETGEN, http://www.hpfem.jku.at/netgen/index.html.

[94] SIMULIA, ABAQUS, http://www.simulia.com/products/abaqus_standard.html.

[95] healthinfotranslations.com, Total hip replacement, http://www.healthinfotranslations.com/pdfDocs/Total_Hip_Replacement_BOS.pdf.

[96] B. A. MacWilliams, D. R. Wilson, J. D. DesJardins, J. Romero, E. Y. Chao, Hamstrings cocontraction reduces internal rotation, anterior translation, and anterior cruciate ligament load in weight-bearing flexion, Journal of Orthopaedic Research 17 (1999) 817–822.

[97] Villoy Implants, http://villoy.com/Villoy%20Implants%20V-200%20V-250%20Hip%20Stem.html.

[98] E. Kuhl, F. Balle, Computational Modeling of Hip Replacement Surgery: Total Hip Replacement vs. Hip Resurfacing, TECHNISCHE MECHANIK 25 (2005) 107–114.

[99] E. F. McCarthy, J. S. Khurana, P. J. Zhang, Essentials in Bone and Soft-Tissue Pathology, Springer, 2009.

[100] V. C. Mow, R. Huiskes, Basic Orthopaedic Biomechanics and Mechano-biology, Lippincott Williams & Wilkins, 2005.

[101] N. Verdonschot, R. Huiskes, Mechanical effects of stem-cement interface characteristics in total hip replacement, Clinical Orthopaedics & Related Research 329 (1996) 326–336.

[102] H. Weinans, R. Huiskes, Trends of mechanical consequence and modeling of a fibrous membrane around femoral hip prostheses, IEEE Computational Science & Engineering 23 (1990) 991–1000.

[103] S. Saha, A. Roychowdhury, Application of the Finite Element Method in Orthopedic Implant Design, Journal of Long-term Effects of Medical Implants 19 [1] (2009) 55– 82.

[104] T. Günter, B. Merz, R. Mericske-Stern, J. Schmitt, R. Leppek, M. Lengsfeld, Testing dental implants with an in vivo finite element model, Biomedizinische Technik. Biomedical engineering 45 (10) (2000) 272–276.

[105] R. v. Liere, J. D. Mulder, J. v. Wijk, Computational steering, Future Generation Computer Systems 12 (1997) 441–450.

[106] A. Borrmann, P. Wenisch, M. Egger, C. van Treeck, E. Rank, Collaborative Computational Steering: Interactive collaborative design of ventilation and illumination of operating theatres, ICE08, Plymouth.

[107] S. G. Parker, D. Beazley, C. R. Johnson, Computational Steering Software System and Strategies, IEEE Computational Science & Engineering 4 (1997) 50–59.

[108] N. Sugano, K. Ohzono, T. Nishii, K. Haraguchi, T. Sakai, T. Ochi, Computed-tomography-based computer preoperative planning for total hip arthroplasty, Computer Aided Surgery 6 (1998) 320–324.

[109] S. Eggli, M. Pisan, M. Miiller, The value of preoperative planning for total hip arthroplasty, Journal of Bone and Joint Surgery 80B (1998) 382–390.

[110] A. M. DiGioia, D. Simon, B. Jaramaz, M. Blackwell, The value of preoperative planning for total hip arthroplasty, Computer Assisted Orthopaedic Surgery Symposium 80B (1995) 382.

[111] D. Bongini, M. Carfagni, L. Governi, A semiautomatic computer program for selecting hip prosthesis femoral components, Computer Methods and Programs in 3 63(2) (2000) 105–115.

[112] C. Dick, J. Georgii, R. Burgkart, R. Westermann, Computational Steering for Patient-Specific Implant Planning in Orthopedics, In Proceedings of Visual Computing for Biomedicine (2008) 83–92.

[113] C. Dick, J. Georgii, R. Burgkart, R. Westermann, Stress Tensor Field Visualization for Implant Planning in Orthopedics, IEEE Transactions on Visualization and Computer Graphics 15(6) (2009) 1399–1406.

[114] M. Riedel, W. Frings, T. H. Eickermann, S. Habbinga, P. Gibbon, D. Mallmann, A. Streit, F. Wolf, T. H. Lippert, Collaborative Interactivity in Parallel HPC Applications, Springer US, 2010.

[115] J. Vetter, K. Schwan, High performance computational steering of physical simulations, Proceedings of the 11th International Symposium on Parallel Processing, IPPS 97 (1997) 128–132.

[116] K. Manjunathachari, K. Satyaprasad, Modeling and simulation of parallel processing architecture for image processing, Journal of Theoretical and Applied Information Technology 3 (2007) 1–11.

[117] A. Düster, M. Rücker, AdhoC^3 – User's Guide, Lehrstuhl für Bauinformatik, Technische Universität München (2001).

[118] A. Düster, H. Bröker, H. Heidkamp, U. Heißerer, S. Kollmannsberger, R. Krause, A. Muthler, A. Niggl, V. Nübel, M. Rücker, D. Scholz, AdhoC^4 – User's Guide, Lehrstuhl für Bauinformatik, Technische Universität München (2004).

[119] H. Effenberger, A. Heiland, T. Ramsauer, W. Plitz, U. Dorn, A model for assessing the rotational stability of uncemented femoral implants, Archives of Orthopaedic and Trauma Surgery 121 (1-2) (2000) 60–64.

[120] A. Niggl, E. Rank, R.-P. Mundani, H.-J. Bungartz, Organizing a p-Version Finite Element Computation by an Octree-Based Hierarchy, in: Proc. of the Int. Conf. on Adaptive Modeling and Simulation, 2005.

[121] B. Chapman, G. Jost, R. van der Pas, Using OpenMP, The MIT Press, 2007.

[122] A. Kiessling, An Introduction to Parallel Programming with OpenMP, http://www.roe.ac.uk/ifa/postgrad/pedagogy/2009_kiessling.pdf.

[123] A. George, Nested Dissection of a Regular Finite Element Mesh, SIAM Journal on Numerical Analysis 10 (1973) 345–363.

[124] M. Bader, C. Zenger, A fast solver for convection diffusion equations based on nested dissection with incomplete elimination, in: Euro-Par 2000 Parallel Processing, 2000, pp. 795–805.

[125] P. Charrier, J. Roman, Partitioning and Mapping for Parallel Nested Dissection on Distributed Memory Architectures, in: Lecture Notes In Computer Science, 1992, pp. 295–306.

[126] Y. Zhuang, J. Canny, Real-time Global Deformations, in: Algorithmic and computational robotics, 2001, pp. 97–105.

[127] C. Hedrick, Introduction to the Internet Protocols, Australian UNIX systems User Group Newsletter 10 (1) (1989) 75 – 76.

[128] K. Siyan, Inside TCP/IP: a comprehensive introduction to protocols and concepts, New Riders Pub, 1997.

[129] R. Tougher, Linux Socket Programming In C++, http://tldp.org/LDP/LG/issue74/tougher.html.

[130] W. R. Stevens, B. Fenner, A. M. Rudoff, UNIX Network Programming: The sockets networking API, Addison-Wesley, 2004.

[131] S. Rai, An Introduction to Socket Programming, http://www.ee.lsu.edu/suresh/prog4710/handout1.pdf.

[132] J. H. Keyak, Y. Falkinstein, Comparison of in situ and in vitro CT scan-based finite element model predictions of proximal femoral fracture load, Medical Engineering and Physics 25(9) (2003) 781–787.

[133] J. H. Keyak, I. Y. Lee, H. B. Skinner, Correlations between orthogonal mechanical properties and density of trabecular bone: Use of different densitometric measures, Journal of Biomedical Materials Research 28(11) (1994) 1329–1336.