IMAGE–BASED BOUNDARY ELEMENT COMPUTATION OF THREE–DIMENSIONAL POTENTIAL PROBLEMS

By

HUI ZHANG

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

Washington State University Vancouver
School of Engineering and Computer Science

August 2008

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of HUI ZHANG find it satisfactory and recommend that it be accepted.

_____

Chair

_____

_____

ACKNOWLEDGMENT

I would like to acknowledge my advisor Dr. Xiaolin (Linda) Chen for her outstanding counsel and encouragement. This work could not have been completed without her dedication and assistance.

IMAGE – BASED BOUNDARY ELEMENT COMPUTATION OF

THREE – DIMENSIONAL POTENTIAL PROBLEM

ABSTRACT

by Hui Zhang, MS
Washington State University Vancouver
August 2008

Chair: Xiaolin (Linda) Chen

Image-based boundary element computation is a computer-aided engineering method for performing simulations based on scanning images of physical objects. In this research, an image-based boundary element computational workflow is developed by tightly integrating the steps of image scanning, mesh regularization and the boundary element method. Mesh quality evaluation and mesh regularization strategies were developed to prepare the scanned images for boundary element computation. Two kinds of potential problems, namely the thermal potential and the bio-potential problems, were investigated to examine the feasibility of the integrated image-based boundary element computation. For thermal potential problems, scan images were collected on objects of large scale from laser scanning and small scale from the micro-CT scanning. Boundary element computation was performed to simulate the heat conduction on the scanned models. Numerical accuracy and computation speed were investigated by comparing the boundary element-based computational scheme with the finite element-based scheme. For bio-potential problems, laser scanning was used to scan geometry information of a human head and a brain from anatomically realistic models. Boundary element computation on bio-electrical potential was performed to inversely compute the cortical potential from simulated Electroencephalography (EEG) measurement on the human scalp. Truncated-Singular Value Decomposition (T-SVD) was implemented to tackle the

solution difficulty caused by ill-conditioned matrices. Parallel computing and block matrix computing were performed to improve the computational speed and the efficiency in computational resource usage. Numerical case studies were conducted to demonstrate the efficiency and accuracy of the image-based boundary element method. Our results show that the image-based boundary element method can be an effective and promising approach for many science research and engineering applications.

TABLE OF CONTENTS

LIST OF FIGURES

Dedication

This thesis is dedicated to my mother, father and my fiancée,

who provided both emotional and financial support.

CHAPTER 1

INTRODUCTION

1.1 Overview

Digital modeling of existing complex freeform objects using scanning techniques has gained lots of attention in recent years [1~4]. Instead of contact measurement and modeling by CAD software, scanning offers non-destructive and non-contact measurement and allows people to reconstruct geometric models either outside or inside of real objects.

Based on the digital modeling, numerical computation can be performed to simulate the mechanical behavior and predict different physical characteristics of the scanned objects. Since significant manual work still remains in the modeling step and simulation step, more efforts have been made to build an integrated workflow of the computation based on the scanned images. It is believed that this will enable real-time mechanical characterization of scanned complex objects, which may benefit the industrial and medical technology, e.g. in simulation-based diagnosis.

1.2 Previous work

Acquired digital modeling using laser scanners has been increasingly used in various reverse engineering, virtual reality applications and traditional design applications [5]. For example, in aesthetic and ergonomic design, digital models of complex arbitrary shapes can be reconstructed from hand-sculptured prototypes with 3D digitizers [6, 7]. In the field of biomedical engineering and ergonomics, nuclear magnetic images and X-ray scanning of the complex geometry of the human tissues and organs are the mainly used non-destructive approaches to gain geometric information inside human body [8~10].

To perform simulation on digital models, many researchers have introduced reconstructed geometries into standard finite element studies (FEM). In these studies, image scanning and FEM computation are two separate processes [11]. Also a time-consuming CAD reconstruction must occur between the scanning and simulation. Meanwhile, computer models reconstructed from high-resolution scan images often contain complex shapes and vast geometric details, which generate a large number of elements in meshing phase. Computational cost becomes a serious challenge for these finite element studies of reconstructed digital models, especially when complex arbitrary shapes and large-size problems are involved.

In some cases, the CAD reconstruction step can be eliminated by translating bitmap information from scan images into hexahedral elements for FEM analysis [12, 13]. These image-based methods have eliminated the time-consuming CAD reconstruction step and made a significant step forward [14~17].

In some other cases, the boundary element method is adopted as a solution for the computation cost caused by complex geometric information. Theoretically, BEM has a higher computational accuracy and efficiency than FEM [18].

In addition, a standard data format, STL, is used to store and transport point cloud information between different types of CAD software. An STL file describes a raw unstructured triangulated surface by the unit normal vector and vertices of the triangles using a three-dimensional Cartesian coordinate system. BEM could take advantage of these existing triangles as triangular boundary elements for simulation.

The inverse problem of the Electroencephalogram (EEG) is a biomedical research area which needs both image scanning and bio-potential computation. In past years, much research was done using FEM and BEM [19, 20]. This field becomes an ideal field to apply and examine the image-based computation technique.

1.3 Objectives of this study

Our research is aimed at developing an integrated imaging and computation solution for reconstructed digital models. At present, the computational process based on the finite element simulation is not efficient enough for industrial application. The development of an integrated workflow based on boundary element simulation can be potentially beneficial for many industrial and biomedical applications.

Specifically, for the study of thermal potential problems, our research objectives include:

- Acquire three dimensional scan images of physical objects based on laser scanning and micro-CT scanning.
- Evaluate the effect of mesh quality on computational accuracy and develop strategies to improve the mesh quality.
- Perform image-based BEM simulation and evaluate its computational accuracy and efficiency, in comparison to the FEM results.

For the study of bio-potential problems, our research objectives include:

- Acquire three dimensional scan images from anatomically realistic models of a human head by laser scanner.
- Validate the image-based boundary element computation by comparing numerical results with theoretically available solutions of simplified multi-layer spherical shell models.
- Conduct image-based boundary element simulation for the EEG inverse problem.
- Implement the Truncated-Singular Value Decomposition technique to tackle ill-conditioned problems in the EEG inverse problem.
- Implement parallel computing and block matrix computing, to alleviate the computational demand on resource usage and time.

1.4 Main work and structure of the thesis

The thesis is structured as follows:

In Chapter 2, some commonly used scanning techniques are reviewed. In particular, the laser scanning and micro-CT scanning are introduced in detail, as they are employed later in this research.

In Chapter 3, the concept of image-based BEM is introduced. Comparing to the FEM, BEM has special advantages in the modeling stage. The integration of scan imaging and boundary element computation results in a more streamlined computational workflow. Strategies for improving the scan imaging results for computation are explained.

In Chapter 4, the image-based BEM is studied for thermal potential problems. Reconstructed models from both laser scanning and micro-CT scanning are included in the thermal studies. The simulation process and results from the BEM are compared with the ones from the FEM in terms of computational accuracy and efficiency.

In Chapter 5, image-based BEM is used to solve the inverse EEG problem, which is a typical bio-potential problem. Due to the signal noise and the large-scale computation, truncated SVD, parallel computing, and block matrix computing are implemented to obtain the solution.

In Chapter 6, numerical results and computing process are discussed, and some conclusions are drawn for the image-based BEM and its applications.

Finally, in Chapter 7 some future research and application areas are discussed.

CHAPTER 2

IMAGE SCANNING METHODS

2.1 Overview

This chapter reviews some commonly used image scanning methods, and focuses mainly on the details about image acquisition and registration are introduced for laser scanning and X-ray scanning, which are employed in this research.

Laser scanning is a non-contact optical method to measure the outside geometry of objects. Image registration is a necessary step after this scanning activity. X-ray scanning methods, such as Computed Tomography (CT), are non-invasive imaging methods to capture the three-dimensional image inside objects. Magnetic resonance imaging (MRI) is a non-invasive method using nuclear magnetic resonance to scan the inside of an object. This is particularly useful with the geometry information of living tissues.

2.2 Laser scanning

Laser scanning is using a scanner to acquire a multitude of x, y, and z coordinates on the surface of a physical object. Each discrete x, y, and z coordinate is referred to as a point. The collection of all these points is referred to as a "point cloud". Typical formats for point cloud data are either a triangular mesh representation of the point cloud in a STL file format or a file containing the coordinate values for each point in an ASCII text format.

The laser scanner used in this research is Konica Minolta Vivid 900 - 3D Laser Scanner, shown in Figure 2-1. It is designed for rapid manufacturing, reverse engineering, performance correlations (FEA/CFD analysis), and other engineering applications.

Figure 2-1 VIVID 900 laser scanner

2.2.1 Point cloud acquisition

The scan process can generate point cloud and STL polygonal mesh images. With the Tele-lens, the physical resolution is 0.039 mm on the object surface. Scanning distance should be limited between 0.6m and 1.2m. In the 'Fine' mode, every single image from one scan contains 307,000 pixels, with a pixel size of 22 um. Each scan takes about 2.5 seconds to complete.

Normally a single scan is just one part of the scanning task. After scanning from different perspectives, image registration and surface defect repair are necessary.

2.2.2 Image registration

As shown below in Figure 2-2, in a scanning procedure, images from four different perspectives are taken in different scans. Each scanned image contains only one side of the object. The scan images from different views were combined into one whole image by reference points on the physical object. This process is normally referred to as Registration, which registers points from different coordinate systems into one common coordinate system.

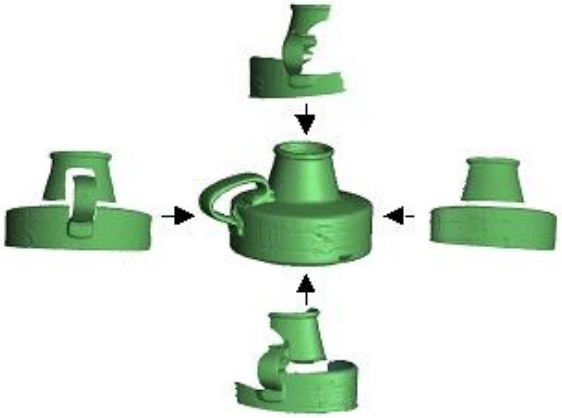


Figure 2-2 Schematic diagram of image registration

2.2.3 Surface defect repairing

After registration, the 3D image usually contains one or more open surfaces. For simulation all surfaces must contain no opening and be continuous. These surfaces are called as ‘water-tight’ surfaces. In addition, due to the influence of unfavorable surface conditions, some surfaces may lose information, contain holes, or generate defects such as self-close bubbles, disconnected parts, or facet intersections. Editing using reverse engineering software, Geomagic Studio, must be done to build water-tight surfaces for numerical simulation.

2.2.4 Alternative of MRI

MRI is a non-invasive method using nuclear magnetic resonance to scan the inside of objects, e.g. a living organ. One key step to solve the EEG inverse problem, which is studied in Chapter 5, is to get three dimensional image of the brain surface. In this research, laser scanning on medical education models is adopted instead of applying a real MRI reconstruction on volunteers.

2.3 X-ray scanning

X-ray is a form of electromagnetic radiation with a wavelength in the range of 10 to 0.01 nanometers. In X-ray image scanning, the image contains information about the intensity reduction inside the three-dimensional object. X-ray absorption difference between materials provides information about interfaces between different materials. In particular, X-rays were found to be able to identify bony structures. This technique has been developed for their use in medical imaging, known as radiology. X-rays are useful in diagnosis mainly of the skeletal system and some soft tissue.

Computed tomography (CT) is a medical imaging method employing X-ray tomography through digital geometry processing. It first captures a series of X-ray microscopic images around a single axis of rotation and then generates a three-dimensional image by combining this series of two-dimensional X-ray images. The three dimensional image reveals the reconstruction inside of an object.

In this research, the SKYSCAN 1074 Micro-CT scanner, shown in Figure 2-3, is used as a compact, non-destructive, three-dimensional microscopy. The maximum scanned area size is 30mm by 30mm. Each scanned image from the X-ray camera contains 768x576 pixels, and the pixel size is 40 um.

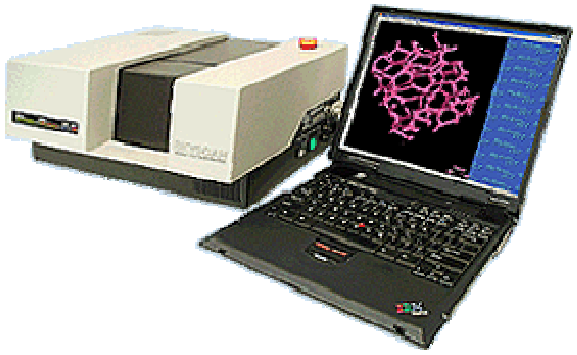


Figure 2-3 SKYSCAN1074 portable Micro-CT scanner

Integrated with the CT scanning, the auto-registration system will generate a water-tight surface automatically. However, for some scanning samples, such as bovine

bone, the X-ray resolution is much higher than the resolution needed in the digital model. Thus there is usually too much data in the point cloud collected from the X-ray scanner, and some data contains unwanted noise. A reverse engineering software package, Geomagic Studio, is often used for scaling, repairing defects and editing the polygon data into acceptable resolution.

CHAPTER 3

IMAGE - BASED BOUNDARY ELEMENT METHOD

3.1 Overview

After imaging, numerical simulation can be performed to find the approximate solutions of partial differential equations (PDEs) which describe physical phenomena on the digitized models. These numerical simulation methods include the Finite Element method (FEM), the Boundary Element Method (BEM), Computational Fluid Dynamics (CFD) or a combination of several methods. Current researches mainly uses either FEM or BEM.

In this chapter, the image-based BEM is introduced and compared with the image-based FEM. A way to improve the boundary mesh for BEM is also constructed and examined.

3.2 Integrated image-based BEM

In the imaging step, a digitizer collects geometric coordinates on the object's surface into a 3D point cloud. After removing erroneous points (i.e., outliers caused by the influence of surface reflectance in laser scanning), a tessellated surface (polygon mesh) can be created from the point cloud through surface triangulation. The end product of the imaging process is in general a polygon surface of the scanned object stored in stereolithography (STL) format. After imaging, FEM or BEM computation can be employed to analyze the digital model.

FEM has been widely used in engineering analysis to find approximate solutions of PDEs as well as of integral equations. When applied for reverse engineering simulation, FEM computation presents an inefficient workflow as shown below. For example, a solid model, represented using non-uniform rational B-spline (NURBS) functions, needs to be reconstructed from the STL data to bridge the gap between

imaging and computation. The computation of image-based FEM relies heavily on this solid model reconstruction, where the complexity of the data transformation involved often requires the use of sophisticated reverse engineering software together with much user intervention. After the solid reconstruction, FEM still needs a three-dimensional meshing step to discretize the solid models into finite elements, of which the order of magnitude increases cubically with the element density on the length scale (element number ~ N^3,where N is the element density on length scale).

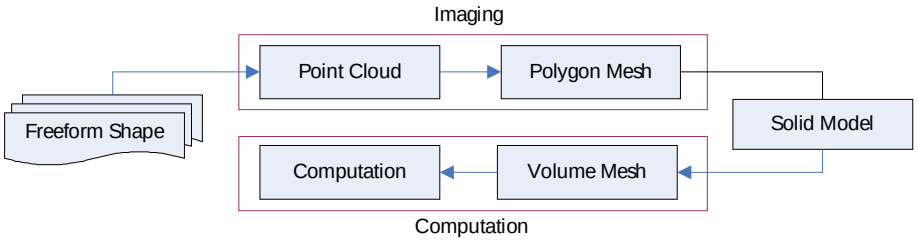


Figure 3-1 Flow Chart for image-based FEM[23]

BEM is a numerical computational method applied in engineering and science including solid mechanics, heat transfer and electromagnetic problems. It can also be employed to analyze the digital model. Different from the FEM, BEM discretizes the surface into boundary elements. The STL file from the digital modeling step allows BEM to use the existing triangles directly as triangular boundary elements. Thus, imaging and computation are joined as one integrated step. In addition, the order of magnitude of boundary element increases quadratically with the element density on length scale (element number ~ N^2). As shown below, the simplified computation workflow makes it beneficial to use BEM instead of FEM on the scanned image.

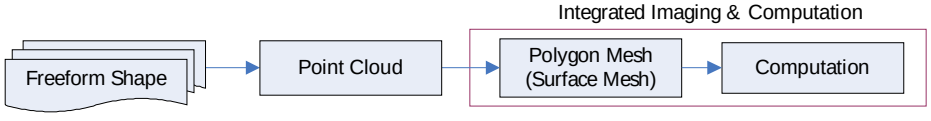


Figure 3-2 Flow Chart for image-based BEM[23]

BEM solves linear partial differential equations by formulating integral equations. The integral equations are exact solutions of the governing partial differential equation. Then the given boundary conditions are used to fit boundary values into the integral equation. The computational accuracy of BEM appears higher than FEM because BEM integral equations are the analytically exact solutions.

In post-processing, the integral equation can be used again to calculate the solution directly at any desired point in the interior of the solution domain. This allows the users to retrieve the field information at interested locations at the post-processing stage. This flexibility is important for realistic applications where the problem size is huge, and yet only surface results of the 3D domain are needed to finish a task.

However the boundary element formulations typically give rise to fully populated matrices. The storage requirements and computational time grow according to the square of the problem size. To improve the computational speed and alleviate the storage limit, parallel computing and block matrix computing can be adopted.

3.3 Mesh quality and mesh regularization

For the BEM computation, the quality of mesh can influence both the numerical accuracy and computation speed. In this section, the quality of boundary element mesh will be quantified by using an element radius-ratio Q factor. Details of the mesh quality evaluation and mesh regularization will be explained next.

3.3.1 Evaluation of mesh quality

Mesh quality control is important in element-based computations because it affects the computation convergence and numerical accuracy. In this image-based study, it is found that the scanned image data do not always come out as high-quality mesh for

BEM computation. To regularize the mesh, we use a simple radius-ratio element shape measure introduced in this section.

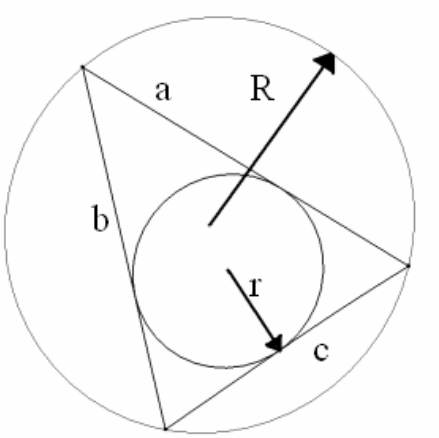


Figure 3-3 Illustration for the mesh quality of a triangular element

The quality factor Q for a triangular element is defined as twice of the ratio between the radius of its inscribed circle and the radius of its circumscribed circle:

$$S = \frac{a+b+c}{2}$$
$$r = \sqrt{\frac{(S-a)(S-b)(S-c)}{S}}$$
$$K = S \cdot r$$
$$R = \frac{a \cdot b \cdot c}{4K}$$
$$Q = 2 \cdot \frac{r}{R}$$

By this definition, the Q factor will fall in the range between 0 and 1. For example, in an equilateral triangle:

$$S = \frac{a+b+c}{2} = \frac{3a}{2}$$
$$r = \sqrt{\frac{(S-a)(S-b)(S-c)}{S}} = \frac{\sqrt{3}}{6}a$$

$$K = S \cdot r = \frac{\sqrt{3}}{4} a^2$$

$$R = \frac{a \cdot b \cdot c}{4K} = \frac{\sqrt{3}}{3} a$$

$$Q = 2 \cdot \frac{r}{R} = 1$$

Similarly, a degenerate element where the three vertices are collinear has a corresponding Q factor of 0, because *S* becomes zero in the calculation.

For BEM computation, a higher Q generally indicates a better element shape. With this measure, ill-shaped elements can be singled out and then treated accordingly.

3.3.2 Influence of mesh quality

To study the effect of mesh quality on the simulation results, a cube with 3072 isosceles right triangular elements was studied for a steady-state heat conduction problem (see figure below). A cube was first created and meshed. Then the mesh was extended along the y-dimension by a factor of 5 and then a factor of 8 times without changing the number of elements. Static heat conduction on these three cases were studied by giving the same material property and same boundary condition (The temperature was set as 0 and 1 on the two faces that are perpendicular to *y* direction, and an adiabatic condition was given on the other faces.)

BEM simulation results are given in contour plots below in Figure3-4, and we found that the numerical error increases significantly while elements are elongated into bad shapes, of which the Q factor is smaller.

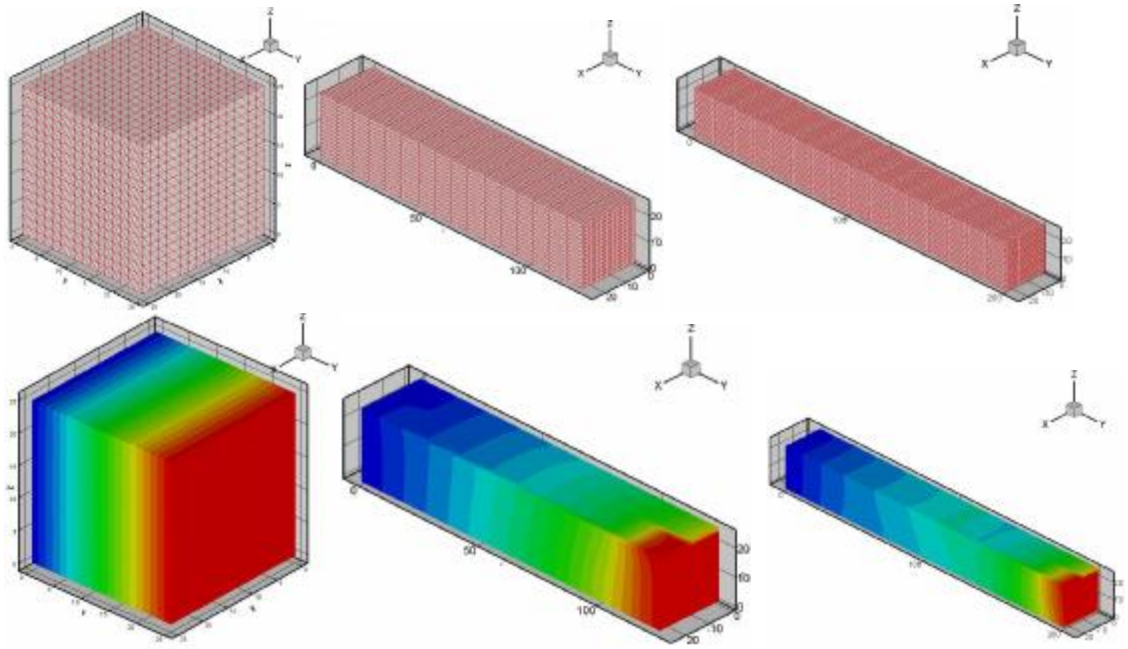


Figure 3-4 Comparison of mesh quality on computation accuracy (not in scale)

In the plot below, numerical results also show a trend that the computation speed slows down dramatically as the element qualities decrease (measured by Q factor).

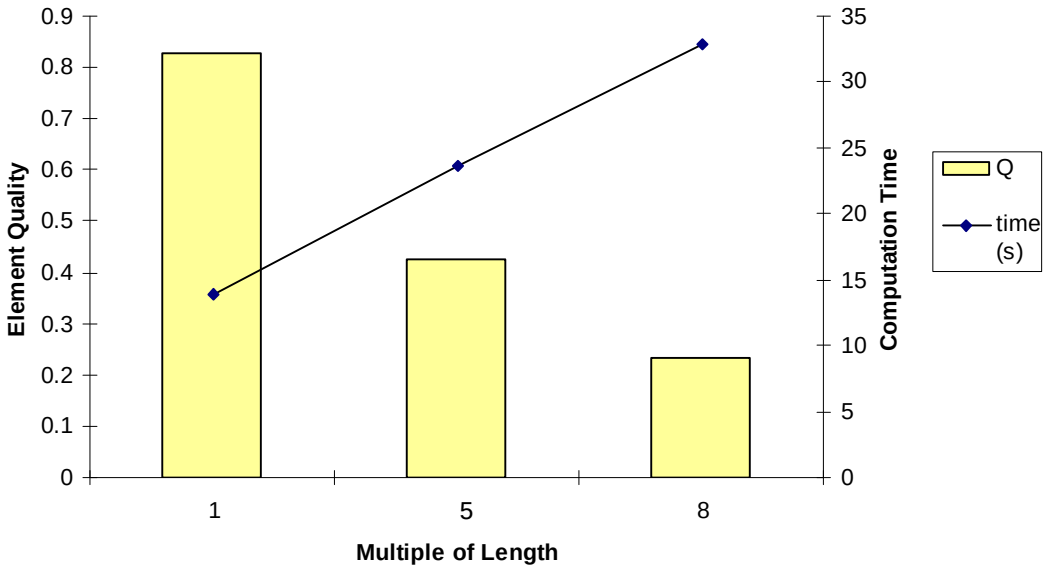
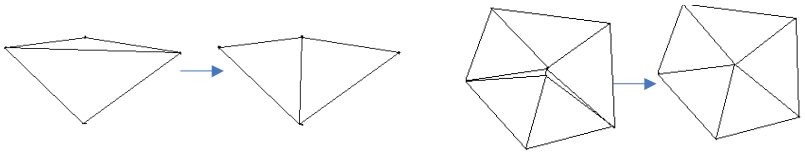


Figure 3-5 Comparison of mesh quality and computation time

3.3.3 Mesh regularization

Based on the research in the previous section, the mesh quality is considered as an important factor influencing both the numerical accuracy and computation speed in BEM simulation. Thus we need methods to regularize mesh and improve the mesh quality before conducting the BEM computation.

For a triangular boundary element, the quality can be improved by either swapping or collapsing. Swapping is used for "cap-like" elements that contain a large obtuse angle. Swapping the collective edge can improve mesh quality of two kinds of triangle elements, as shown in Figure 3-6(a). Collapsing is used for "needle-like" elements that contain a small acute angle. Collapsing the degenerating edge into a vertex can remove a pair of needle-like elements and improve average mesh quality, as shown in Figure 3-6(b).



(a) Swapping (b) Collapsing

Figure 3-6 Two methods to regularize elements of bad quality[23]

For the mesh regularization on the entire mesh, the simplified procedure presented in Figure 3-7 is used. This flowchart takes the input of a prescribed mesh quality control factor and a geometric data set stored in STL format. Unqualified elements with Q factors under the given control factor (usually around 0.3) are treated either by edge collapsing or by edge swapping. After iteration, the regularized node position and element connectivity information can work as a BEM mesh and are still stored in STL format.

This workflow was programmed in MATLAB. Details about the program are given in the Appendix A1.

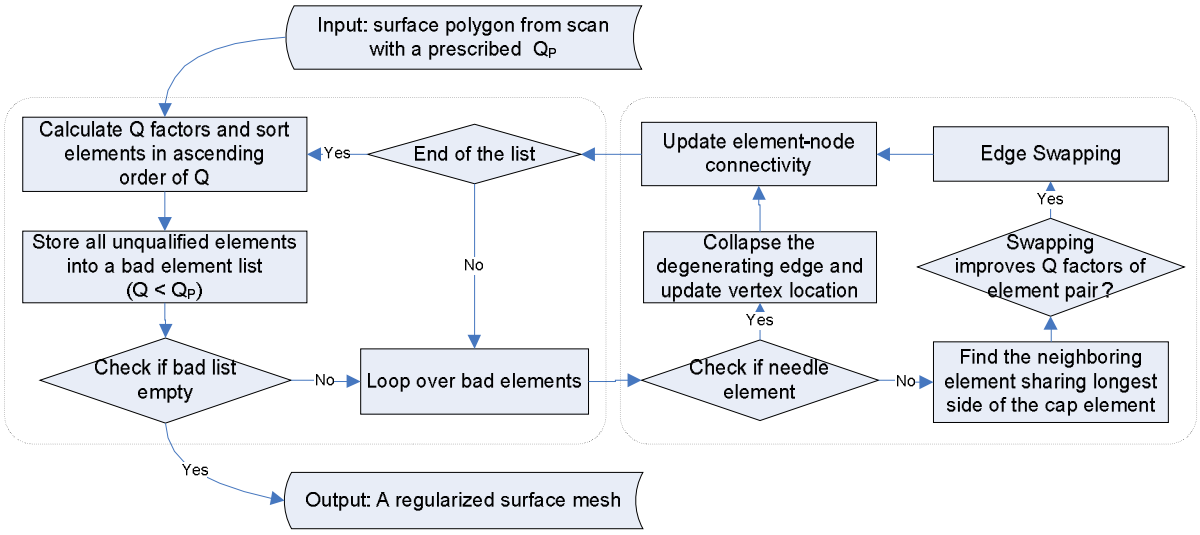


Figure 3-7 Workflow for mesh regularization iteration

CHAPTER 4

IMAGE – BASED BEM FOR THERMAL POTENTIAL PROBLEMS

4.1 Overview

In this chapter, the image-based BEM is formulated for thermal potential problems and performed on different models. The simulation process after the scanning step is designed to be an automated computational workflow. Thermal potential problems are chosen here because they are one of the easiest CAE computations, in which only one degree of freedom (DOF), temperature, need be solved. The integrated work flow will apply to any other BEM simulation, such as stress analysis or CFD.

4.2 BEM formulation for thermal potential problems

In this section, we follow classic techniques in describing the BEM formation, taking 3D steady-state heat conduction as an example [22, 23]. In BEM, the governing partial differential equations are transformed into integral representation, referred to as the boundary integral equations (BIEs). The problem is then solved based on the discretized BIEs over a domain's boundary. The problem dimension is generally reduced by one in BEM. In other words, only surface discretization is needed for 3D problems [21]. Also the governing equations are exactly satisfied at each field point so that it can provide more accurate solutions, even when using a fairly coarse boundary mesh.

For 3D steady-state heat conduction, assuming no internal heat source, the temperature potential field ϕ must satisfy the following Laplace equation:

$$\nabla^2\phi = 0 \tag{1}$$

To establish the BIEs, we consider the Green's function (also referred to as the fundamental solution) at a field point y in an infinite medium due to a unit heat source at point x. The Green's function satisfies the following equation:

$$\nabla^2 \boldsymbol{G}(\boldsymbol{x},\boldsymbol{y}) + \delta(\boldsymbol{x},\boldsymbol{y}) = 0 \tag{2}$$

Where $\delta(\boldsymbol{x},\boldsymbol{y})$ is the Dirac δ-function representing a unit concentrated heat source; $\boldsymbol{G}(\boldsymbol{x},\boldsymbol{y})$ is the Green's function given by

$$\boldsymbol{G}(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{4\pi \boldsymbol{r}}$$

for a 3D potential problem, with r representing the distance between the source point x and the field point y.

Applying the Gauss theorem, we obtain the following identity (or a reciprocal relation) involving the potential field ϕ and the fundamental solution:

$$\int_{\boldsymbol{V}} [\boldsymbol{G}(\boldsymbol{x},\boldsymbol{y})\nabla^2\phi(\boldsymbol{y}) - \phi(\boldsymbol{y})\nabla^2\boldsymbol{G}(\boldsymbol{x},\boldsymbol{y})]d\boldsymbol{V} = \int_{\boldsymbol{S}} [\boldsymbol{G}(\boldsymbol{x},\boldsymbol{y})\frac{\partial\phi(\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})} - \phi(\boldsymbol{y})\frac{\partial \boldsymbol{G}(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})}]d\boldsymbol{S}(\boldsymbol{y}) \tag{3}$$

where n(y) is the surface normal at a field point y.

Substituting equations (1) and (2) into (3), we derive an integral representation for the potential field:

$$\phi(\boldsymbol{x}) = \int_{\boldsymbol{S}} \left[\boldsymbol{G}(\boldsymbol{x},\boldsymbol{y})\frac{\partial\phi(\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})} - \phi(\boldsymbol{y})\frac{\partial \boldsymbol{G}(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})}\right] d\boldsymbol{S}(\boldsymbol{y}), \forall \boldsymbol{x} \in \boldsymbol{V} \tag{4}$$

Here, x is an arbitrary source point inside domain V, and y an arbitrary field point on the domain's boundary S. A domain potential is thus related to some integral of surface potentials and surface fluxes through equation (4).

Now we define heat flux q as

$$\boldsymbol{q}(\boldsymbol{y}) = \frac{\partial\phi(\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})}$$

and introduce

$$\boldsymbol{F}(\boldsymbol{x},\boldsymbol{y}) = \frac{\partial \boldsymbol{G}(\boldsymbol{x},\boldsymbol{y})}{\partial \boldsymbol{n}(\boldsymbol{y})} = -\frac{1}{4\pi \boldsymbol{r}^2}\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{n}(\boldsymbol{y})}.$$

If we let the source point x in a domain V approach the boundary S, then we will have the following boundary integral equation (BIE):

$$\boldsymbol{c}(\boldsymbol{x})\phi(\boldsymbol{x}) = \int_{\boldsymbol{S}} \left[\boldsymbol{G}(\boldsymbol{x},\boldsymbol{y})\boldsymbol{q}(\boldsymbol{y}) - \boldsymbol{F}(\boldsymbol{x},\boldsymbol{y})\phi(\boldsymbol{y})\right] \boldsymbol{dS}(\boldsymbol{y}), \forall \boldsymbol{x} \in \boldsymbol{S} \tag{5}$$

where c(x) is a constant coefficient depending on the smoothness of the boundary at a point x (e.g., =0.5, for smooth surface). At this point, both the source point x and the field point y are located on the boundary surface S now.

To subtract the kernel singularity existing in the BIE, we apply a special loading case with constant $\phi(\boldsymbol{y})$ and zero $\boldsymbol{q}(\boldsymbol{y})$ (similar to a rigid body motion for elasticity) to Equation (5), and the coefficient term can be expressed as:

$$\boldsymbol{c}(\boldsymbol{x}) = -\int_{\boldsymbol{S}} \boldsymbol{F}(\boldsymbol{x},\boldsymbol{y}) \boldsymbol{dS}(\boldsymbol{y}), \forall \boldsymbol{x} \in \boldsymbol{S} \tag{6}$$

Substituting Equation (6) into (5), we derive the following form of BIE:

$$\int_{\boldsymbol{S}} \boldsymbol{F}(\boldsymbol{x},\boldsymbol{y})\left[\phi(\boldsymbol{y}) - \phi(\boldsymbol{x})\right] \boldsymbol{dS}(\boldsymbol{y}) = \int_{\boldsymbol{S}} \boldsymbol{G}(\boldsymbol{x},\boldsymbol{y})\boldsymbol{q}(\boldsymbol{y}) \boldsymbol{dS}(\boldsymbol{y}), \forall \boldsymbol{x} \in \boldsymbol{S} \tag{7}$$

Equation (7) is a non-singular BIE form. The singularity in G kernel can be eliminated by using a polar coordinate transformation ($\boldsymbol{ds} = \boldsymbol{rdrd}\theta$), and the singularity in F kernel can also be removed after using a one-term Taylor's series expansion of the density function (temperature ϕ) together with the polar coordinate transformation.

After discretizing the boundary S into elements with nodes, we can write the BIE at each node. Applying the boundary conditions and constraints, the BIEs can be rearranged into a linear equation system:

$$\mathbf{Az} = \mathbf{b} \tag{8}$$

where A is the coefficient matrix, b is the known load vector and z the unknown vector.

The coefficient matrix A represents the thermal interaction between any two node points. The final linear system of equations collected from all surface nodes is then solved simultaneous to obtain the unknown temperatures or heat fluxes on the boundary. Although BEM relies solely on the surface discretization, accurate information in the

interior domain can be readily obtained from equation (4), once the surface information is obtained from equation (8).

4.3 The image-based BEM with laser-scanning

Image-based BEM with laser-scanning is performed first. Applications of image-based BEM with laser scanning may include digital model product design improvement, ancient building evaluation, etc.

4.3.1 Laser Scanning

The scan started after the objective was located on the center of the scan window, as shown below. The laser scanner acquired the geometry on the surface of a physical object in terms of a point cloud with a multitude of (X, Y, Z) coordinates. A photo was taken for each view and used to assist the image registration.

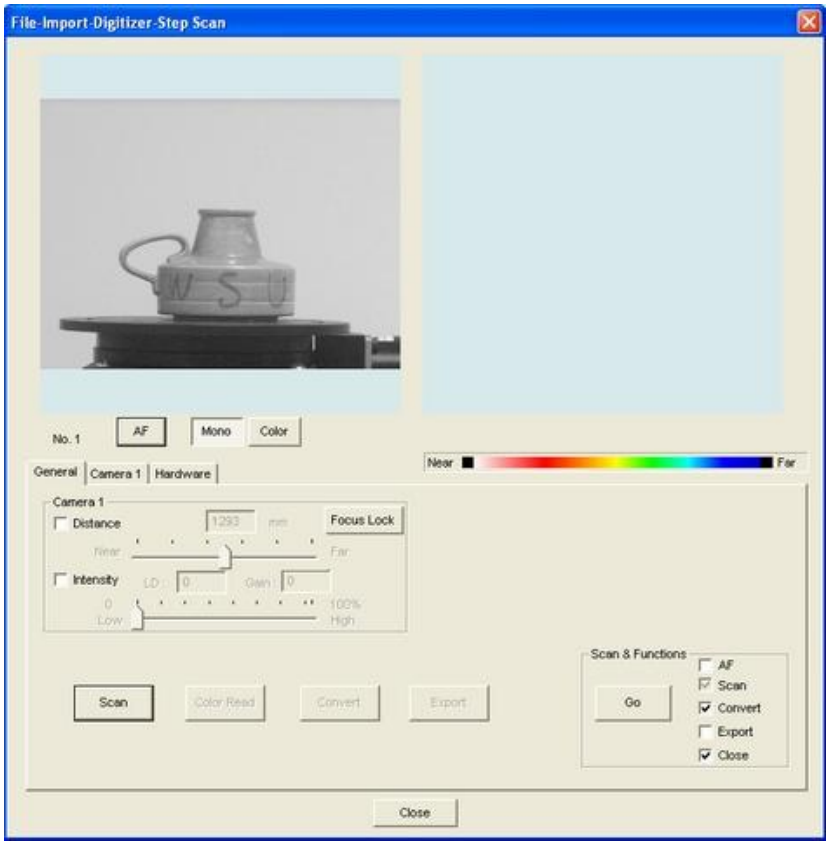


Figure 4-1 Scanning of a lamp

As shown below, in the registration step, pairs of reference points are used to combine two scanned image together. Typically obvious features such as sharp corners are easy to use as reference points. For this smooth and even-textured lamp, we added seven letters 'WSUENCS' to provide reference points by the corners, ends and crossings of letters.

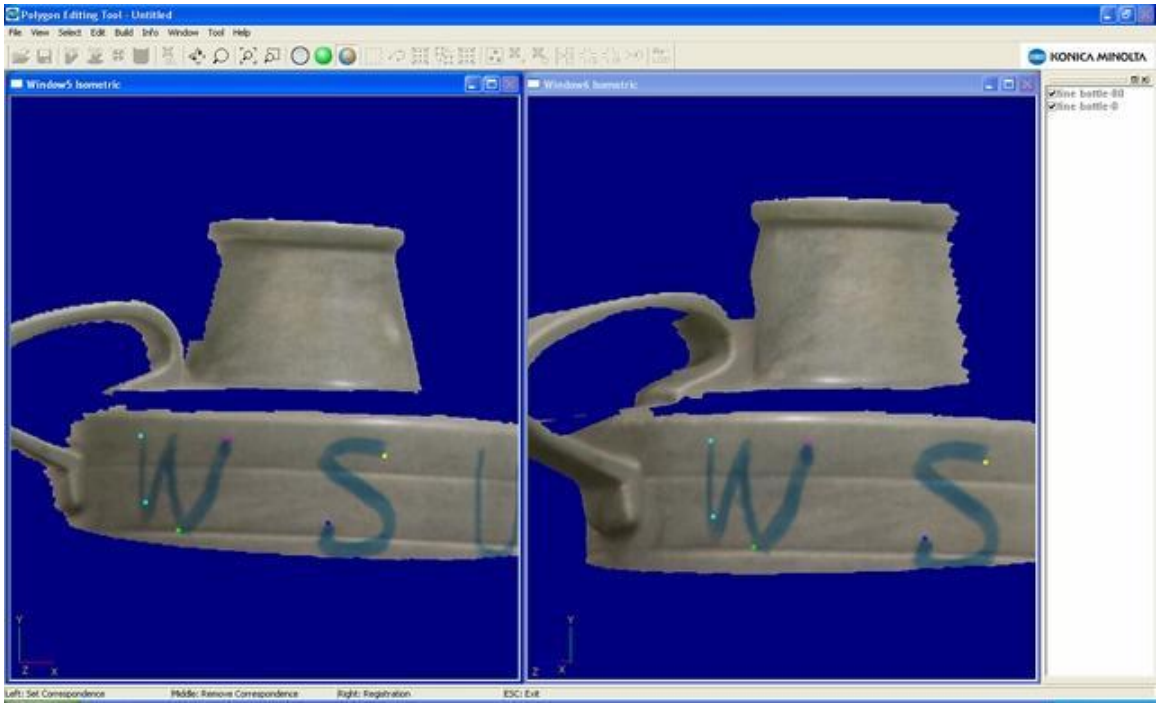


Figure 4-2 Image registration using five couples of reference points

Using the reverse engineering software GeoMagic, defects on the combined surface were treated properly. The figures below show an example of hole selecting and filling.

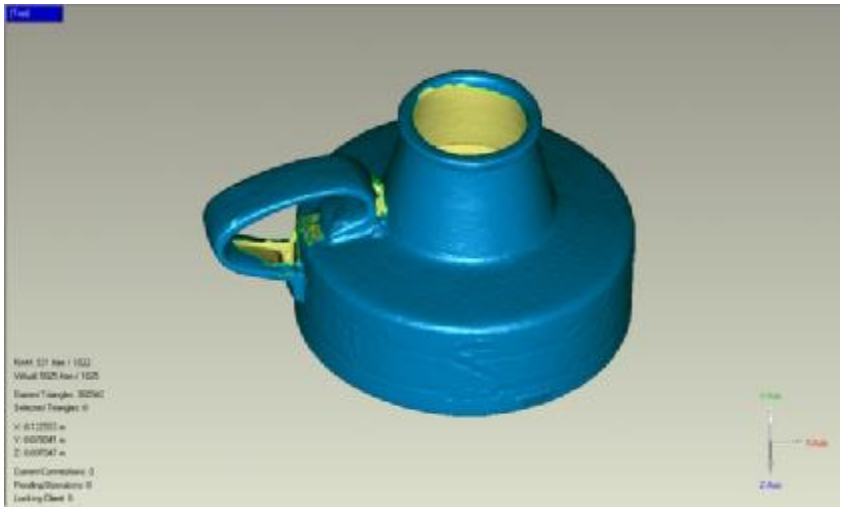


Figure 4-3 Selecting holes on the reconstructed surface

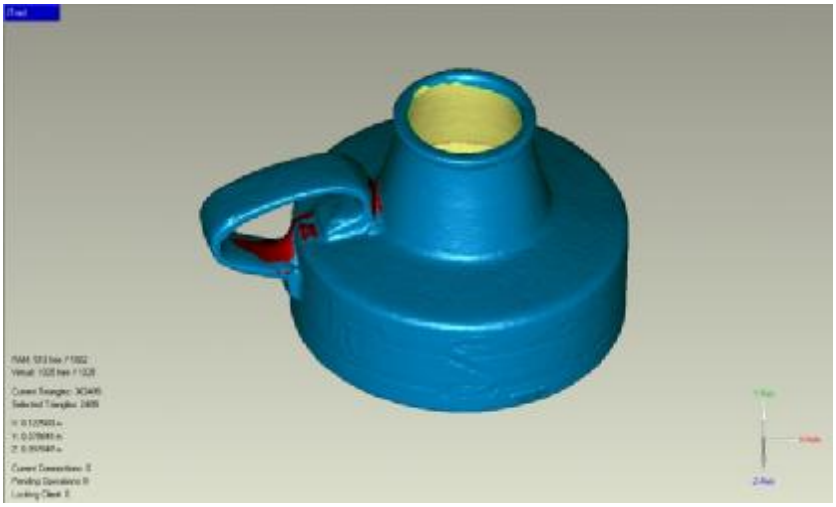
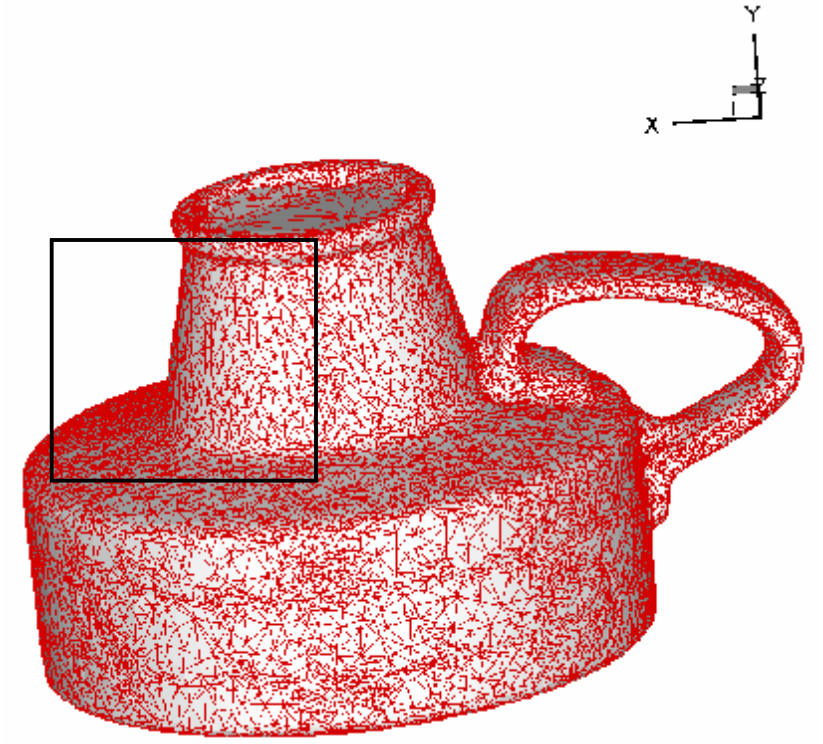


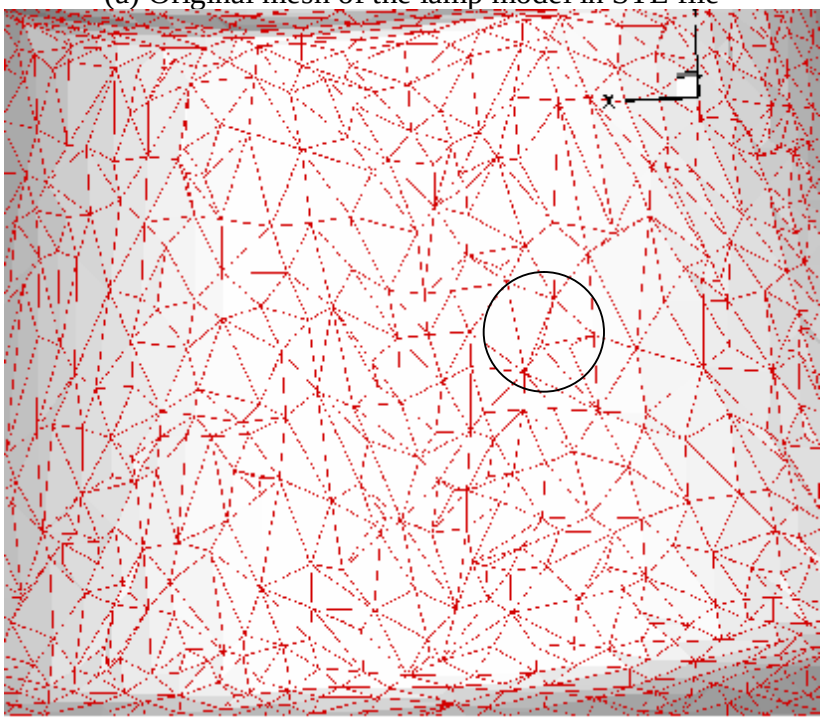
Figure 4-4 Filling holes on the reconstructed surface

4.3.2 Mesh regularization

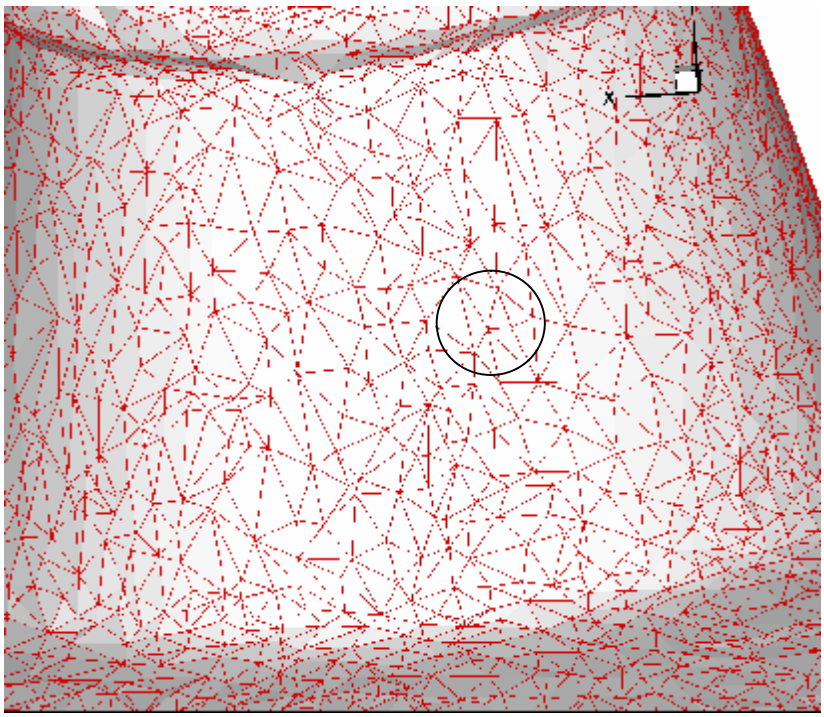
The lamp model contained 30000 triangular elements. A control factor of 0.3 was specified in the regularization procedure, and elements with Q factors lower than the control factor were regularized. After mesh regularization, the number of element became 28184. The triangular elements within the block mark area before and after mesh regularization are magnified and shown below. Circles are used to highlight two pairs of element which are regularized by swapping.



(a) Original mesh of the lamp model in STL file



(b)Elements within the marked area before mesh regularization



(c) Elements within the marked area after mesh regularization

Figure 4-5 Mesh plots before and after mesh regularization

To evaluate the mesh regularization quantitatively, the mesh improvement is measured by the change of Q factor distributions, as shown in Figure 4-6. The solid line and the dashed line present the element Q factor before and after regularization, respectively. In this case, the elements with Q factors lower than 0.3 was significantly reduced after regularization, and those with Q in between 0.6 and 1 were accordingly increased.

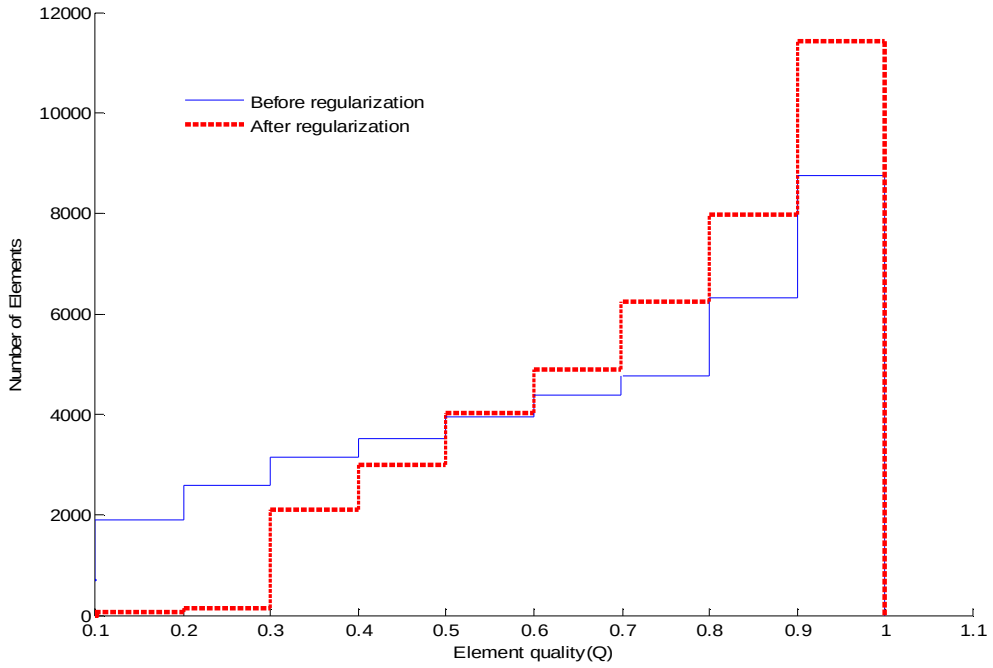
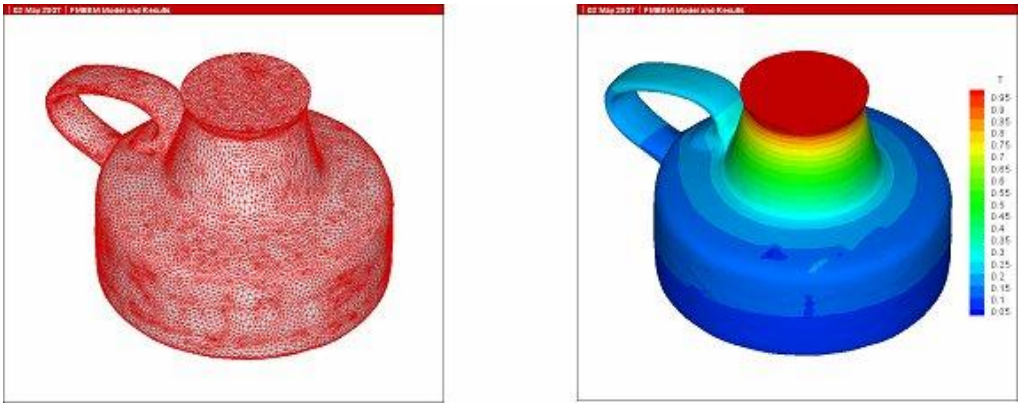


Figure 4-6 Element quality distribution before and after mesh regularization

4.3.3 Heat conduction computation by BEM

The regularized BEM mesh is shown in Figure 4-7 (a). A heat transfer simulation was performed on this model. The top of the model was set at a temperature of 1 ^{o}C, while the bottom's temperature was 0 ^{o}C. The side surfaces were adiabatic. The entire object was assumed to be a solid piece made of isotropic and homogeneous material, which was assigned a constant thermal conductivity of 1 W/m$\cdot$ ^{o}C.

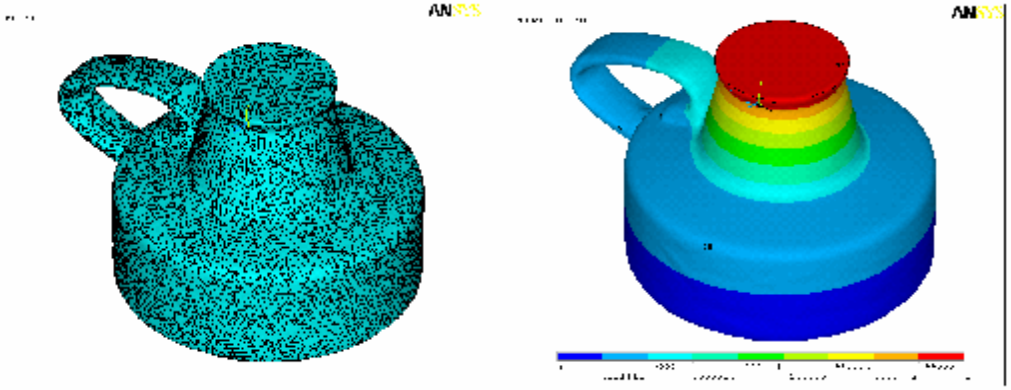


(a) Surface mesh (b) Temperature plot from BEM computation

Figure 4-7 BEM computation on the regularized mesh

4.3.4 Heat conduction computation by FEM

As a comparison, a lamp model of the same material property and boundary conditions was studied using ANSYS, a highly optimized FEM package. The simulation result of temperature contours is shown below. Results obtained from the developed BEM fits well with the ANSYS results.



(a) Volume mesh (b) Temperature plot from ANSYS

Figure 4-8 Temperature plot from FEM computation

The surface mesh used for the BEM computation contains 42,810 triangular elements, while the solid mesh for ANSYS, which is a highly optimized commercial FEM package, uses 403,271 tetrahedral elements to maintain the same surface mesh

density. Both simulations were run on the same desktop PC with a 3.2 GHz Pentium IV processor and 1.5 GB memory. The recorded CPU time was close to 1 hour (3593 seconds) for ANSYS and less than 15 minutes (885 seconds) for the accelerated BEM simulation. As expected, the developed BEM showed a computational advantage over the highly optimized commercial code ANSYS by significantly reducing the problem size and complexity and therefore the computational cost for such simulation.

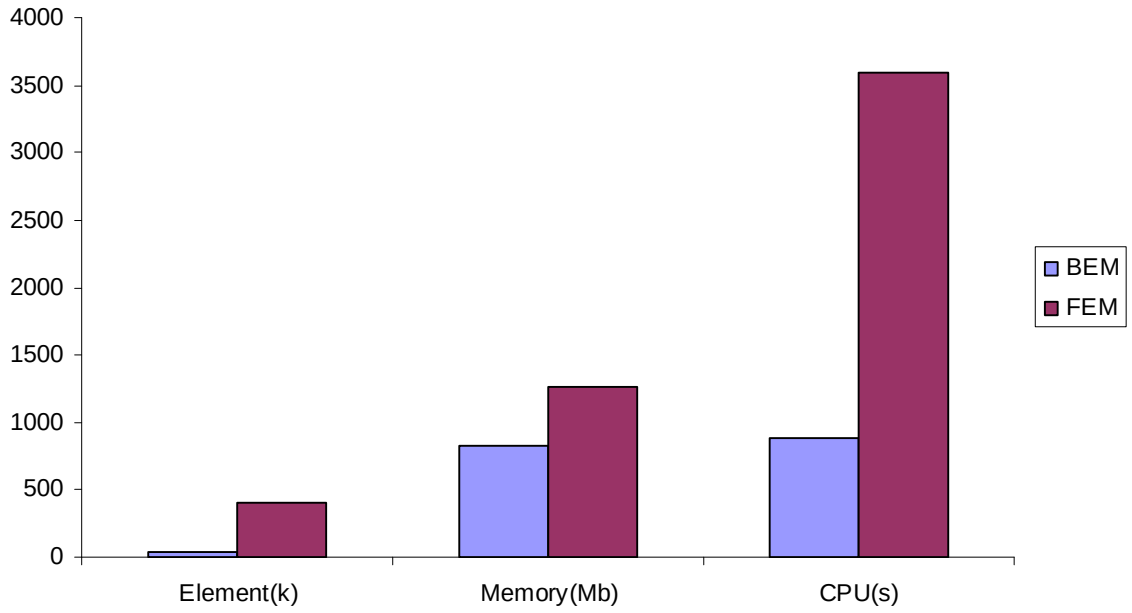
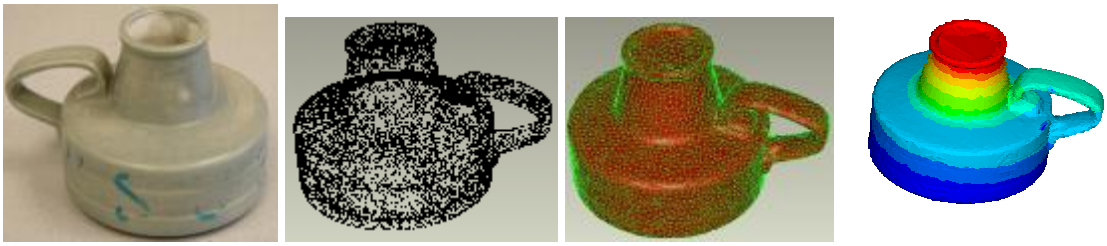


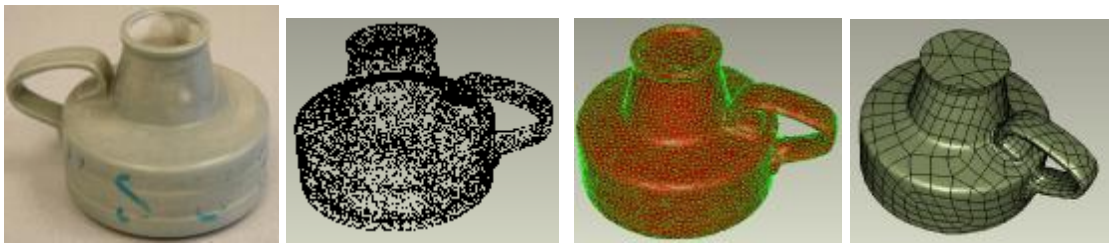
Figure 4-9 Comparison between BEM and FEM on the number of elements, memory requirement and computation time

As discussed in Chapter 3, the BEM has potential advantage on the modeling process, and later research further justified this point. As shown in Figure 4-10, the image-based BEM simulation uses the geometric information stored in STL format as a triangular surface mesh directly for computation. The FEM simulation has to generate patches, grids and NURBS surfaces based on the STL surfaces (Figure 4-11 (d) ~ (f)). Then it must build a volume in the FEM software (ANSYS) and mesh it using three dimensional elements (Figure 4-11 (g)).

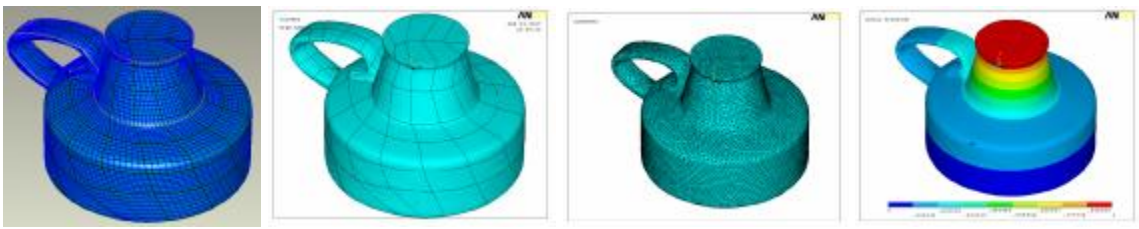


(a) Object (b) Point cloud (c) Polygons mesh (d) BEM result

Figure 4-10 Data flow for image-based BEM



(a) Object (b) Point cloud (c) Polygon mesh (d) Patches



(e) Grids (f)NURBS (g) Volume mesh (h) FEM result

Figure 4-11 Data flow for image-based FEM

4.4 Image-based BEM with micro-CT scanning

Image-based BEM with micro-CT scanning is examined next. The image-based BEM with micro-CT scanning can be used for many biomedical and material science applications.

4.4.1 Computed tomography (CT)

X-ray scanning is used with the auto-registration system to generate a water-tight surface automatically, as shown in Figure 4-12. The scaling and surface defect repairs were done in GeoMagic.

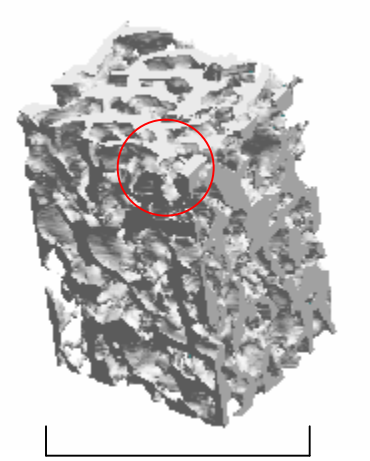


Figure 4-12 Automatically registered image data of bovine bone sample

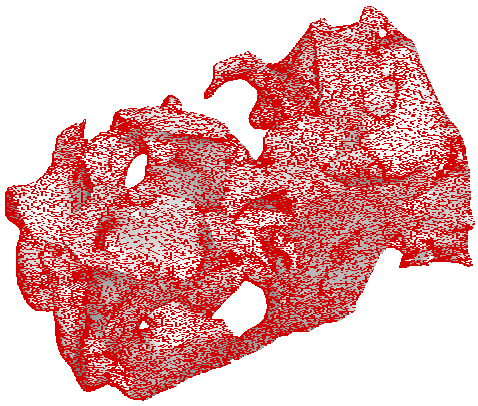


Figure 4-13 BEM mesh stored in the STL file

(Zone-in view of the marked area in Figure 4-12)

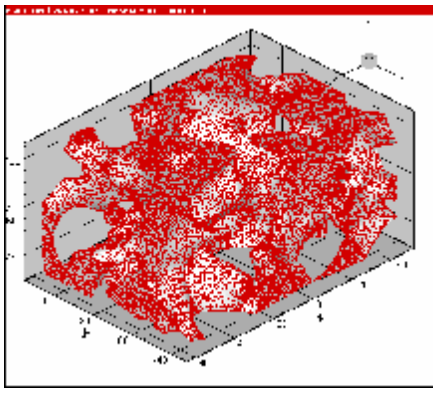
4.4.2 Heat conduction computation

Thermal analyses of the X-ray scanned microstructure of bone models were performed using the developed BEM to evaluate its capability in handling large scale problems with more complex geometry.

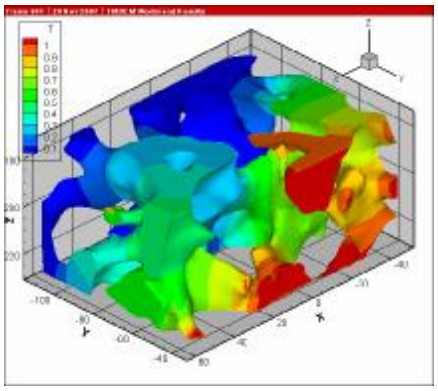
Surface polygon of the acquired digital models was imported for the image-based BEM analysis. Solutions were successfully obtained on a desktop PC (3.2 GHz Pentium IV processor and 1.5 GB memory). Figure 4-13 shows the BEM meshes and thermal results for bone microstructures. About 120,000 and 200,000 triangular elements were used for the BEM meshes (a) and (c) respectively.

A heat transfer simulation was performed on this model. The front surface of the object was set at a temperature of 1 ^{o}C, and the opposite surface's temperature was 0 ^{o}C. The other surfaces were adiabatic. The bone tissue was assumed to be a solid piece made up of isotropic and homogeneous material, which was assigned a constant thermal conductivity of 1 W/m$\cdot$ ^{o}C.

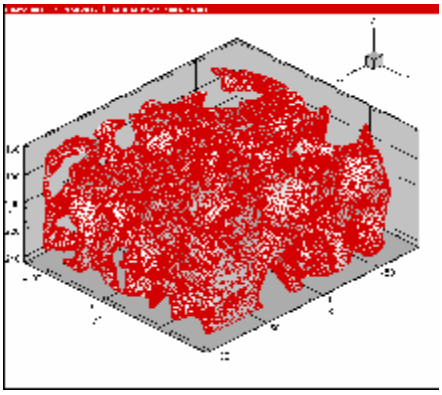
The image-based BEM nicely captured the heat flow from one end to the other when both ends were held at constant temperatures of 0^{o}C and 1^{o}C, respectively.



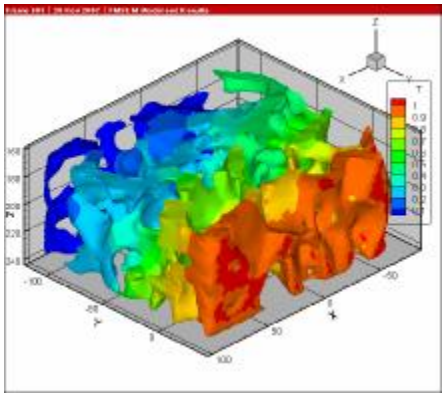
(a) Mesh of 120,000 elements



(b) Temperature results from BEM



(c) Mesh of 200,000 elements



(d) Temperature results from BEM

Figure 4-14 Study for bone samples of different element numbers

The CPU time consumed by the BEM is plotted in Figure 4-16 for three different bone samples. The CPU time increased almost linearly with the problem size for the developed BEM. Roughly 0.6, 1.3 and 3.4 hours were spent on the desk PC to obtain results for the three micro-structural models containing about 70k, 120k and 200k triangular elements, respectively. To achieve similar accuracy, the FEM would generally require model with significantly increased problem size (by 10 to 100 fold) and hence

would take a much longer solution time. These preliminary results demonstrate the effectiveness of the developed BEM, which could be efficient yet not limited to applications where only boundary wall (both exterior and interior) information is needed.

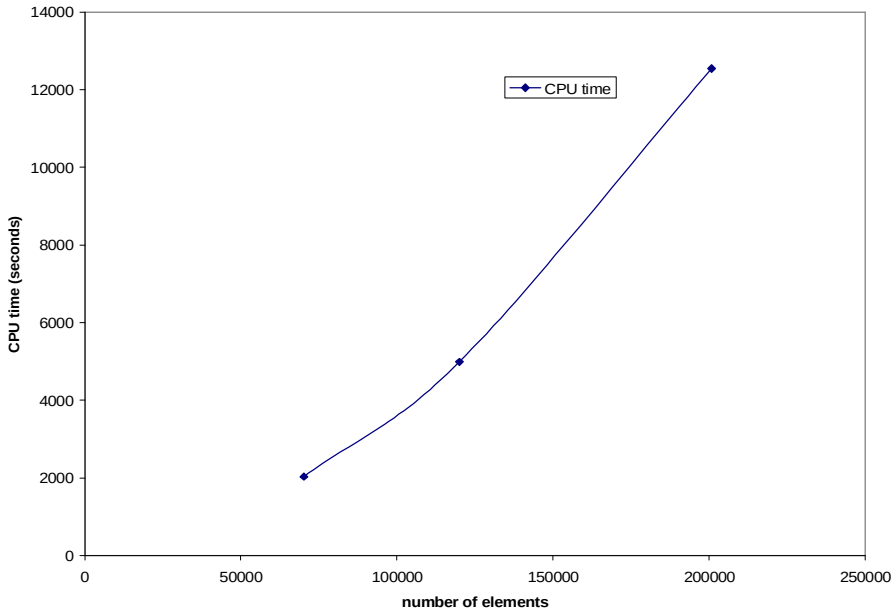


Figure 4-15 BEM computation time versus the problem size

A FEM study is also tried using ANSYS. However, the memory of desktop computer is not enough even in the volume meshing step. This also indicates the limit of image-based FEM on objects of complex geometry.

CHAPTER 5

IMAGE – BASED BEM FOR BIOELECTRICAL POTENTIAL PROBLEMS

5.1 Overview

Bioelectric potential problems, also known as bio-potential problems, solve for electrical source information based on the electrical measurements at the surface of the skin of a living organism. For bio-potential problems, the main difficulties come from the ill-posed boundary conditions. Also, the stability from signal noise during measurement must be considered.

In this chapter, the image-based BEM is used to solve the Electroencephalogram (EEG) inverse problem, one specialized example from the general field of bio-potential problems. The image-based BEM can be applied to solve the inverse problem by finding the electrical potential solution on the scanned image from human body and then computing the transfer matrix between cortical potentials and scalp potentials. Due to the large computation scale of realistic models, block matrix computing and parallel computing were used to increase computation speed. Results from different numerical cases were used to evaluate the computation process.

5.2 EEG and EEG inverse problem

5.2.1 Electroencephalogram

Electroencephalogram (EEG) is the measurement of brain electrical activity obtained by attaching electrodes on the scalp and recording the measured electrical signal. EEG causes no external physical damage while measuring the brain activities, and it is sensitive over the time domain. Because of these advantages, EEG is widely used in clinics for mental disease diagnosis and related research.

Although EEG is sensitive to the temporal change of potential signal, the accuracy of spatial potential resolution is still limited because the resolution doesn’t

increase with the number of attached electrodes. Researchers in mechanical engineering, biomedical engineering, medical science and especially electrical engineering are looking for a computational method to get a higher resolution [19, 20, 25, 26].

The forward problem of EEG is to computes the scalp surface potentials from the known potential on the cortex surface or the equivalent current dipoles. Boundary conditions are given on both the scalp surface outside (first order derivative of potential along outward normal direction $\partial p/\partial n=0$) and brain surface inside (potential p=p (x)). This problem is classified as an electrical field problem of the third kind of boundary condition, for which it is relatively easy to solve numerically. The EEG forward problem is a necessary step towards the EEG inverse problem and has important applications in biomedical simulations [20].

5.2.2 Inverse problems of EEG

More realistic problems are the inverse problem, which compute the potential on the cortex surface by the measurement of scalp potentials [25, 26]. The difficulty of the inverse solution comes from the non-unique solution of EEG inverse problems. In these problems, which are called ill-posed problems, the boundary conditions ($\partial\Phi/\partial n$ and p) are given only on the scalp surface.

In some studies, either FEM or BEM is applied as a simulation tool to solve the inverse problem. For FEM, mesh generation is reported as the main difference in performing realistic model simulation [20]. In addition, various numerical algorithms are typically needed with the simulation to eliminate numerical error and influence from noise signal [25, 26].

When BEM is used to solve the EEG forward problem, the transfer matrix must be determined before multiplying by the potential array of cortical surface, which results in the potential array on the scalp surface. If an inverse matrix, or pseudoinverse matrix, of the transfer matrix in forward problem is found, the inverse solution can be determined

as well, as the cortical potential can be solved by multiplying this (pseudo)inverse matrix and the scalp potential. A frequently used method to solve the (pseudo)inverse matrix is the Truncated Single-Value Decomposition (TSVD) method.

5.3 BEM formulation for bio-potential problems

In this study, the BEM is used to construct the transfer matrix between the cortical surface and the scalp surface in a simplified model using materials that are all homogeneous and isotropic.

5.3.1 BEM for a shell volume

In this formulation, the BEM is used to solve for the electrical potential for a homogeneous isotropic volume surrounded by a close outer surface and a close inner surface.

For the volume V inside of surface S, Green's second identity can be written as [21, 25]:

$$\iiint_V (A\nabla^2 B - B\nabla^2 A)dV = \iint_S (A\nabla B - B\nabla A)\cdot \mathbf{n} dS \tag{1}$$

where $\mathbf{n}$ is the unit surface normal to surface S at each point (infinitesimal surface element dS). A and B are two scalar functions of position with continuous second derivatives within V.

If the material of V is isotropic and there is no electrical current source existing within V, a formula can be determined by defining A as the scalar electrical potential u and B as $\frac{1}{r}$, where r is the distance from the observation point $\mathbf{r}^*$ located within V to the infinitesimal surface element dS. The formula can be given as[16]:

$$u(\mathbf{r}^*) = \frac{1}{4p}\iint_S u \cdot d\Omega + \frac{1}{4p}\iint_S \frac{1}{r}\cdot\frac{\partial u}{\partial r_n} dS \tag{2}$$

where

$u(\vec{r}^*)$ is the electrical potential at the observation point $\vec{r}^*$;

$d\Omega$ is the solid angle of an infinitesimal dS as seen from $\vec{r}^*$;

$\frac{\partial u}{\partial r_n}$ is the first derivative of potential u with respect to the outward normal to dS.

Assuming a volume is defined by its outer surface Sx and inner surface Sy, (2) becomes:

$$u(\vec{r}^*) = \frac{1}{4p}\iint_{Sx} u \cdot d\Omega + \frac{1}{4p}\iint_{Sx} \frac{1}{r} \cdot \frac{\partial u}{\partial r_n} dS - \frac{1}{4p}\iint_{Sy} u \cdot d\Omega - \frac{1}{4p}\iint_{Sy} \frac{1}{r} \cdot \frac{\partial u}{\partial r_n} dS \qquad (3)$$

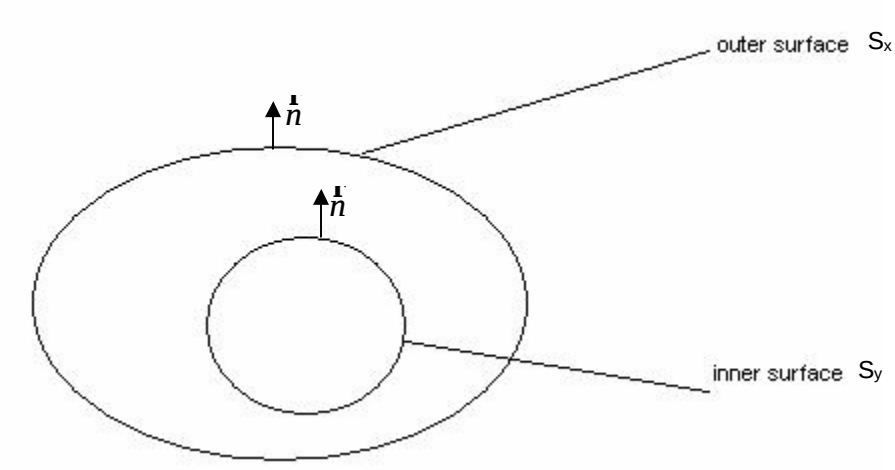


Figure 5-1 A volume between its outer and inner surfaces

By discretizing into triangular elements and taking the limit of approaching the elements on surface, (3) becomes:

$$u_x^i = \frac{1}{4p}\sum_{j=1}^{Nx} u_x^j \cdot \Omega_x^{ij} + \frac{1}{4p}\sum_{j=1}^{Nx} (\frac{\partial u}{\partial r_n})_x^j \cdot g_{xy}^{ij} - \frac{1}{4p}\sum_{j=1}^{Ny} u_y^j \cdot \Omega_y^{ij} - \frac{1}{4p}\sum_{j=1}^{Ny} (\frac{\partial u}{\partial r_n})_y^j \cdot g_{xy}^{ij} \qquad (4)$$

where

$i = 1, 2, ..., Nx$

u_k^i is the potential value at the i th triangular element on surface S_k ,here k notes the surface number;

Ω_k^{ij} is the solid angle subtended by the j th triangular element on surface S_k as seen from the i th triangular element on surface Sx ;

N_k is the number of discretized triangular elements on S_k

$g_{xy}^{ij} = \iint_{\Delta j} \frac{1}{r_{ij}} dS$, where Δj is the j th triangular element on surface Sy , and r_{ij} is the distance between the j th triangular element on surface Sy and the i th triangular element on surface Sx .

Combining the left-hand side of (4) with the first term in the right-hand side, the formula can be rewritten in matrix format:

$$F_{xx}U_x + G_{xx}\Gamma_x + F_{xy}U_y + G_{xy}\Gamma_y = 0 \tag{5}$$

where

U_k is the column vector consisting of potentials at every element on S_k ;

Γ_k is the column vector consisting of $\frac{\partial u}{\partial r_n}$ at every element on S_k ;

F_{jk} and G_{jk} are coefficient matrices with dimensions of N_j by N_k .Calculations of F_{jk} and G_{jk} are given by Banerjee [21]:

$$\begin{cases} F_{jk} = \sum_{n=1}^{3} [\arctan(\frac{z \cdot l_2}{D \cdot r_2}) - \arctan(\frac{z \cdot l_1}{D \cdot r_1}) + sign(z)a\,] \\ G_{jk} = \sum_{n=1}^{3} \{-D \cdot \log(\frac{r_1 + r_2 + L}{r_1 + r_2 - L}) + z \cdot [\arctan(\frac{z \cdot l_2}{D \cdot r_2}) - \arctan(\frac{z \cdot l_1}{D \cdot r_1})] + |z|a\,\} \end{cases} \tag{6}$$

where n is the edge number in a triangular element. Other variables are noted in the figure below:

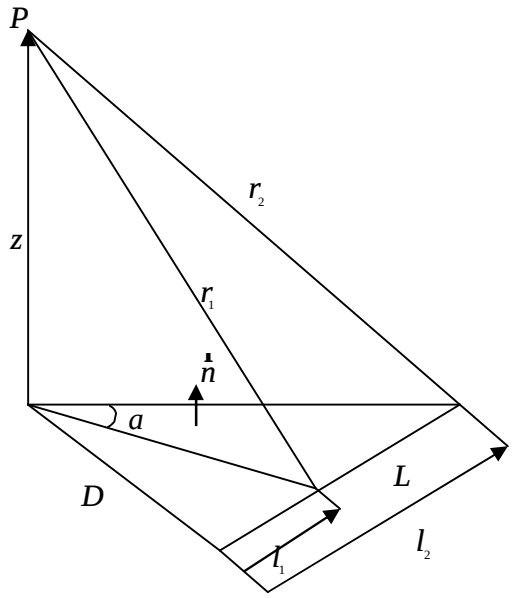


Figure 5-2 Local parameter of coefficient matrices calculation

Similarly, considering the observation point at inner surface from inside of the volume, we have

$$F_{yx}U_x + G_{yx}\Gamma_x + F_{yy}U_y + G_{yy}\Gamma_y = 0 \tag{7}$$

In a forward problem, Γ_x and U_y are used as boundary conditions. Γ_y and U_x are unknowns to be solved in the problem.

Solving (5) and (7) leads to the solution:

$$\begin{cases} U_x = (G_{xy}G_{yy}^{-1}F_{yx} - F_{xx})^{-1}[(F_{xy} - G_{xy}G_{yy}^{-1}F_{yy})U_y + (G_{xx} - G_{xy}G_{yy}^{-1}G_{yx})\Gamma_x] \\ \Gamma_y = (F_{yx}F_{xx}^{-1}G_{xy} - G_{yy})^{-1}[(G_{yx} - F_{yx}F_{xx}^{-1}G_{xx})\Gamma_x + (F_{yy} - F_{yx}F_{xx}^{-1}F_{xy})U_y] \end{cases} \tag{8}$$

, where the superscript -1 indicates the matrix inversion.

In an inverse problem, Γ_x and U_x are used as boundary conditions. Γ_y and U_y are unknowns to be solved in the problem.

Solving equations (5) and (7) leads to the solution:

$$\begin{cases} U_y = (G_{xy}G_{yy}^{-1}F_{yy} - F_{xy})^{+}[(F_{xx} - G_{xy}G_{yy}^{-1}F_{yx})U_x + (G_{xx} - G_{xy}G_{yy}^{-1}G_{yx})\Gamma_x] \\ \Gamma_y = (F_{xy}F_{yy}^{-1}G_{yy} - G_{xy})^{+}[(G_{xx} - F_{xy}F_{yy}^{-1}G_{yx})\Gamma_x + (F_{xx} - F_{xy}F_{yy}^{-1}F_{yx})U_x] \end{cases} \tag{9}$$

, , where the superscript + indicates the pseudoinverse of a matrix.

The initial reason to use a pseudoinverse matrix instead of the inverse matrix is that the matrix $(G_{xy}G_{yy}^{-1}F_{yy} - F_{xy})$ has dimensions of N_x by N_y. When $N_x \neq N_y$, the inverse matrix is not available.

Later in the numerical experiment, the matrix $(G_{xy}G_{yy}^{-1}F_{yy} - F_{xy})$ is found to be ill-conditioned, as it has a large condition number. Thus even $(G_{xy}G_{yy}^{-1}F_{yy} - F_{xy})^{-1}$ is available, its numerical error is still unacceptable. Instead, the pseudoinverse matrix $(G_{xy}G_{yy}^{-1}F_{yy} - F_{xy})^{+}$ computed by SVD method is adopted to decrease the error.

5.3.2 BEM of a multi-shell model

To solve the EEG inverse problem, we assumed the human head was simplified as three volumes V1, V2, and V3 (representing scalp skin, skull, and cerebrospinal fluid) isolated by four surfaces S1, S2, S3, and S4. Γ_1 and U_1 are given on S1 as boundary conditions. The BEM is used to solve for the potential U_4. Due to the limit on the number of measurement electrodes and the requirement of high resolution on brain surface, the number of elements on different surfaces could be different, usually with $N_4 > N_1$.

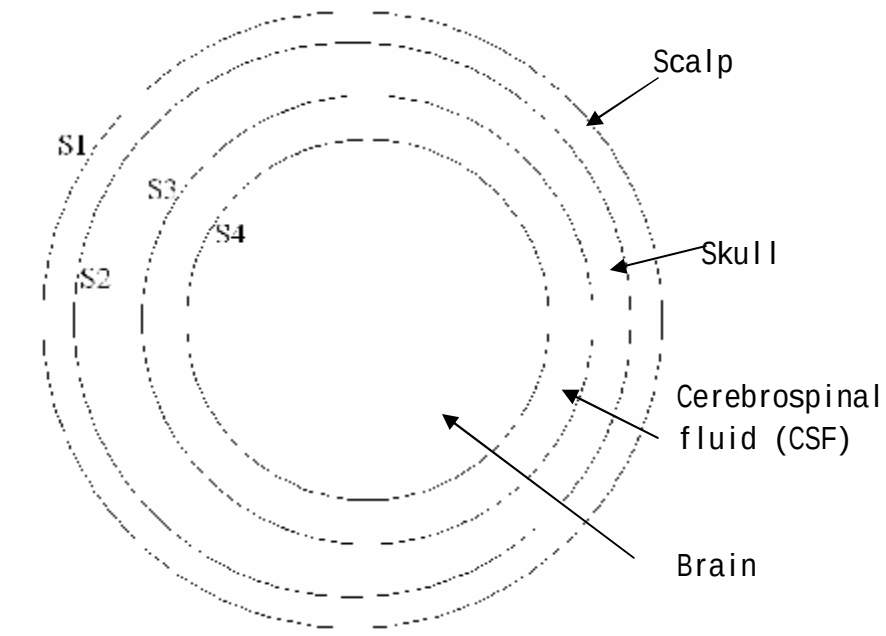


Figure 5-3 A three-shell volume model

In this study, the inverse solution uses Γ_1 and U_1 as boundary conditions to solve for Γ_2 and U_2 , by BEM computation on volume V_1. Then the Γ_2 and U_2 become boundary conditions for BEM computation volume V_2. This process continues until the U_4 is finally solved.

5.4 Truncated-Singular Value Decomposition (Truncated-SVD)

As introduced in section 5.2.2, pseudoinverse matrices must be solved from ill-conditioned matrices or non-square matrices. From previous research [22], we found that three least square methods can be applied on this problem: 1) normal equations; 2) QR decomposition; 3) SVD method. After a literature review and initial investigations on these three methods, the truncated SVD method was selected.

The singular value decomposition (SVD) is a factorization of a rectangular real or complex matrix. Its applications include computation of the inverse matrix, least squares fitting of data and matrix approximation. The truncated SVD method is a reduced version of the full SVD.

The original matrix $T_{m\times n}$ is first decomposed by SVD:

$$[U \ \Sigma \ V^t] = SVD(T_{m\times n})$$

where matrix sizes of U , Σ, and V^t are $m\times m$, $m\times m$,and $m\times n$ respectively. Here V^t means the transpose matrix of V .

We then define $\Sigma' = \Sigma^{-1}$, and set all diagonal elements of Σ' except the r smallest diagonal elements as zeros, where r is the truncation level in SVD procedure. The rest of the matrix is discarded. This entire process is named as truncated SVD. The pseudoinverse matrix T^* of $T_{m\times n}$ can be given by

$$T^* = V \times \Sigma' \times U^t$$

In this research project, truncated SVD is used for all the matrices of which the condition number is larger than 5,000.

5.5 Block matrix computations

In matrix theory, a block matrix is a partition of a matrix into rectangular smaller matrices called blocks [22]. A block partitioned matrix sum, difference and product can be formed involving operations only using the sub-matrices. By block matrix computation, the sub matrices become small enough to be used. This algorithm can also reduce the number of multiplications. However, data operations must be more frequently to read and recorded on the computer hard disks.

5.5.1 Matrix Addition

Assuming A_{ij}, B_{ij} and C_{ij} are all m-by-n matrices, while A, B and C are $2m$-by-$2n$ matrices, then the matrix addition $C = A + B$ can transform to the following format:

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} + \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$C_{11} = A_{11} + B_{11}$$

$$C_{12} = A_{12} + B_{12}$$

$$C_{21} = A_{21} + B_{21}$$

$$C_{22} = A_{22} + B_{22}$$

Each matrix can be divided into 4 sub matrices, each of which occupies one quarter of the original matrix. Matrix addition can be conducted in the procedure above. Accordingly, the computer memory usage decreases to one quarter of the direct addition algorithm.

5.5.2 Matrix Multiplication

In the method below, we assume A_{ij}, B_{ij}, C_{ij}, A, B and C are l-by-m , m-by-n , l-by-n , $2l$-by-$2m$, $2m$-by-$2n$, $2l$ - by-$2n$ matrices. A matrix multiplication of $C = A \times B$ transforms as:

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Sub matrices are calculated and stored as a temporary matrix T_i , using intermediate steps:

$$T_1 = A_{11} + A_{22}$$
$$T_2 = B_{11} + B_{22}$$
$$T_3 = T_1 T_2$$
$$T_4 = A_{21} + A_{22}$$
$$T_5 = T_1 B_{11}$$
$$T_6 = B_{12} - B_{22}$$
$$T_7 = A_{11} T_6$$
$$T_8 = B_{21} - B_{11}$$
$$T_9 = A_{22} T_8$$
$$T_{10} = A_{11} + A_{12}$$
$$T_{11} = T_{10} B_{22}$$
$$T_{12} = A_{21} - A_{11}$$
$$T_{13} = B_{11} + B_{12}$$
$$T_{14} = T_{12} T_{13}$$
$$T_{15} = A_{12} - A_{22}$$
$$T_{16} = B_{21} + B_{22}$$
$$T_{17} = T_{15} T_{16}$$

Finally C matrix can be calculated as:

$$C_{11} = T_3 + T_9 - T_{11} + T_{17}$$
$$C_{12} = T_7 + T_{11}$$
$$C_{21} = T_5 + T_9$$
$$C_{11} = T_3 + T_7 - T_5 + T_{14}$$

This multiplication method is also called Strassen multiplication. It can save about much memory because the block matrix computation requires memory only for three sub-matrices.

5.6 Parallel computing using multi-computers

In this study, the time cost of the largest computational case would be as long as one week. Thus, a faster arithmetic becomes necessary to speed up the real computation. The limit on memory also requires an alterative arithmetic. Parallel computing can be used to speed the computation up and reduce memory requirement by carrying computations out simultaneously.

There are different forms of parallel computing: instruction-level parallelism, data parallelism, task parallelism and hardware supports parallelism. The distributed computing, which use parallel computers, is a good fit for this research project since it was easy to take advantage of the spare computers in the ENCS CAD lab.

Our original computation was written for serial computing, which is run on a single computer having a single Central Processing Unit (CPU). The problem is broken into a discrete series of instructions, and the instructions are executed one after another. Thus, only one instruction may execute at any moment in time. For example, the coefficient matrices F_{jk} and G_{jk} in formula (5) of section 5.3.1 would be computed one by one.

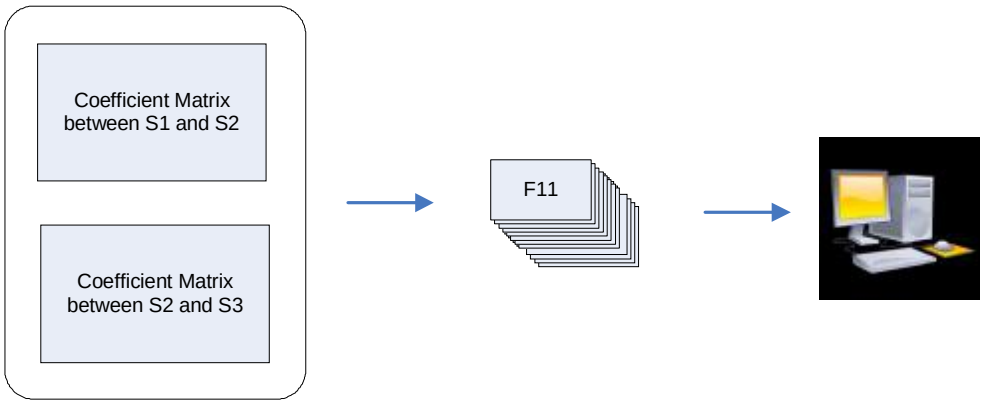


Figure 5-4 BEM computation using serial computing

Parallel computing uses multiple computers to solve a computational problem simultaneously. The problem is broken into discrete parts and each part is further broken down to a series of instructions, with instructions from each part executing concurrently on different computers. Thus the coefficient matrices F_{jk} and G_{jk} in formula (5) of section 5.3.1 can be computed at the same time.

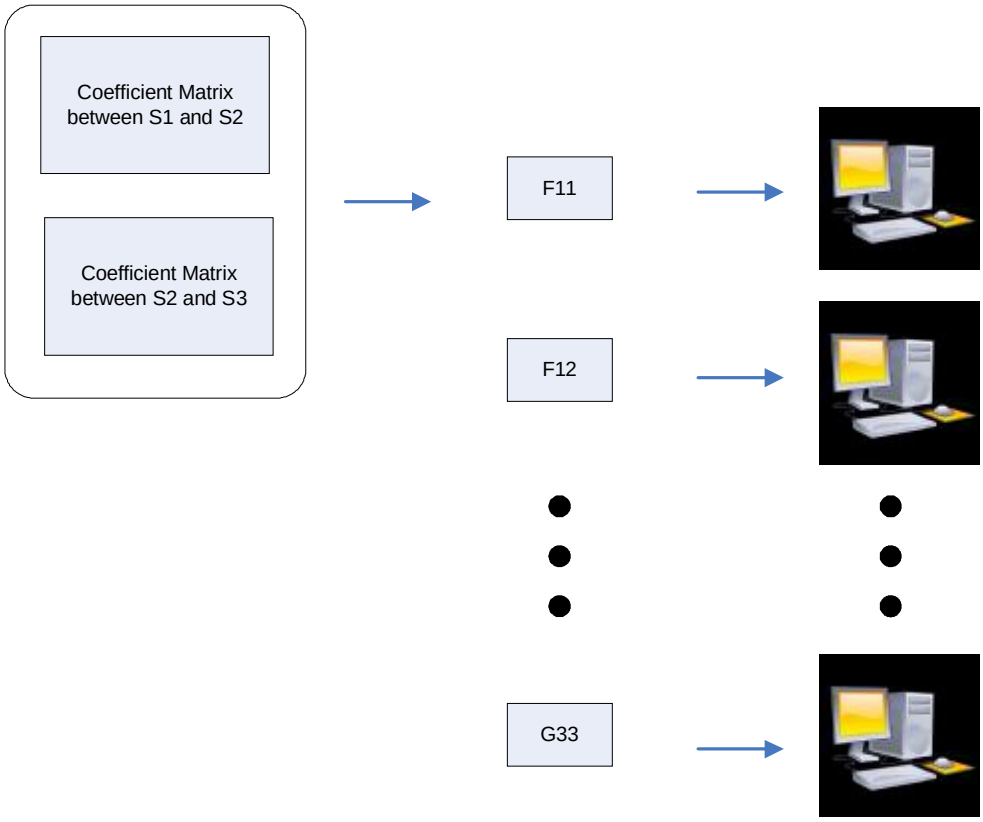


Figure 5-5 BEM computation using parallel computing

For EEG inverse problems, BEM computation is very easily divided into parts because the transfer matrices of different volumes don't influence each other.

In parallel computing, the total computation time stays the same or is perhaps a little longer due to the initialization of multiple tasks. However, the practical computational time (wall clock time) is reduced. In addition, the memory threshold on a single computer becomes lower.

5.7 Surface modeling using laser scanning

In the EEG inverse problem, the interfaces between the scalp, bone, neurolymph and the cortex are the key information needed for simulation. In medical research, a MRI is used to find the interface between different issues or organs.

However, for our research, the primary objective is to check the feasibility of numerical simulation and the workflow of image-based BEM. Costly MRI measurement is replaced by laser scanning, because outputs of these two measurements are both surface information. Accordingly, the real human body is replaced by educational models from the nursing department at WSU Vancouver.

Using a 3D laser scanner, high-resolution images of human head structures are captured from anatomically realistic educational models. Detailed surface representations of the head, brain and skull structures were then reconstructed from the scan images and refined in Geomagic Studio. As shown below, the reconstructed surfaces were assembled into a multi-layer digital human head model. Four volumes can then be defined from the four consecutive surface layers, with different conductivity parameters set for each volume.

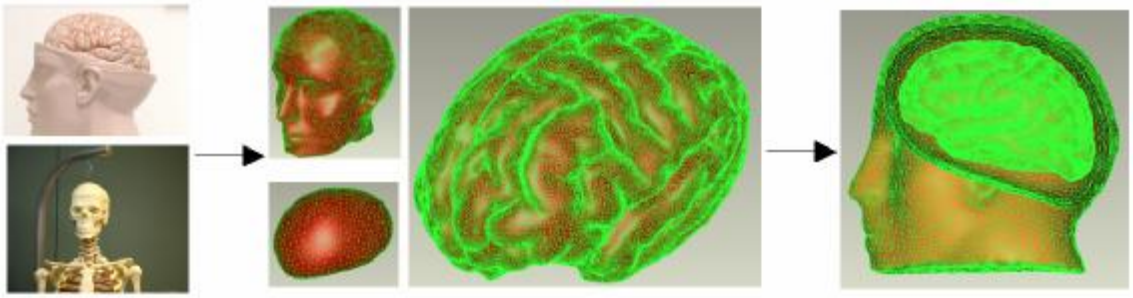


Figure 5-6 Image reconstruction using laser scanning and reverse engineering software

5.8 Simulations on the spherical models

Computations using spherical models were performed before using the realistic models, as simpler spherical models can more easily be used to compare the BEM inverse solution with the theoretical solution to estimate the numerical error. The spherical models are also used to examine the influence of white noise in signal.

5.8.1 Theoretical formula of the spherical model

To examine the computational accuracy and compare numerical errors of different computation cases, an accurate theoretical solution is necessary as a reference.

The formula below is used to compute the theoretical potential produced by a dipole in a homogeneous conducting sphere to examine the numerical result. Due to the linear property of electrical field, this formula can also be used for multi-dipole condition.

The electrical potential *P* generated by a dipole in a homogeneous sphere can be given by [24]:

$$P_{\vec{r}} = \frac{\vec{P}}{4ps}\left\{\frac{\vec{r}-\vec{r}_o}{r_p^3} + \frac{\left(\vec{r}-\frac{r^2}{R^2}\vec{r}_o\right)}{R^3 r_{pi}^3} + \frac{1}{R^3 r_{pi}}\left[\vec{r} + \frac{\vec{r}\,\frac{r_0 r}{R^2}\cos j - \frac{r^2}{R^2}\vec{r}_o}{r_{pi}+1-\frac{r_o r}{R^2}\cos j}\right]\right\}$$

where $P_{\vec{r}}$ is the potential on any location $\vec{r}$ within the sphere, $\vec{P}$ is dipole vector, $\vec{r}_o$ is the location of dipole, R is the radius of the outer sphere, $\cos j$ is the cosine of the angle between $\vec{r}$ and $\vec{r}_o$, and $r_{pi} = \frac{(r^2 + r_o^2 - 2rr_o\cos j)^{1/2}}{R}$.

In our test cases, the theoretical calculation and BEM simulation were conducted on a concentric four- shell homogeneous spherical model. The radii of spherical surfaces S1, S2, S3, and S4 are 37, 35, 32, and 30 respectively. Each surface is made up by 200 triangular elements. A unit dipole along y-direction [0, 1, 0] is posed in the origin of this coordinate system, which is also the centre of spheres. The contour plots of theoretical

calculations on S1, the most outside surface, and S4, the most outside surface, are shown below.

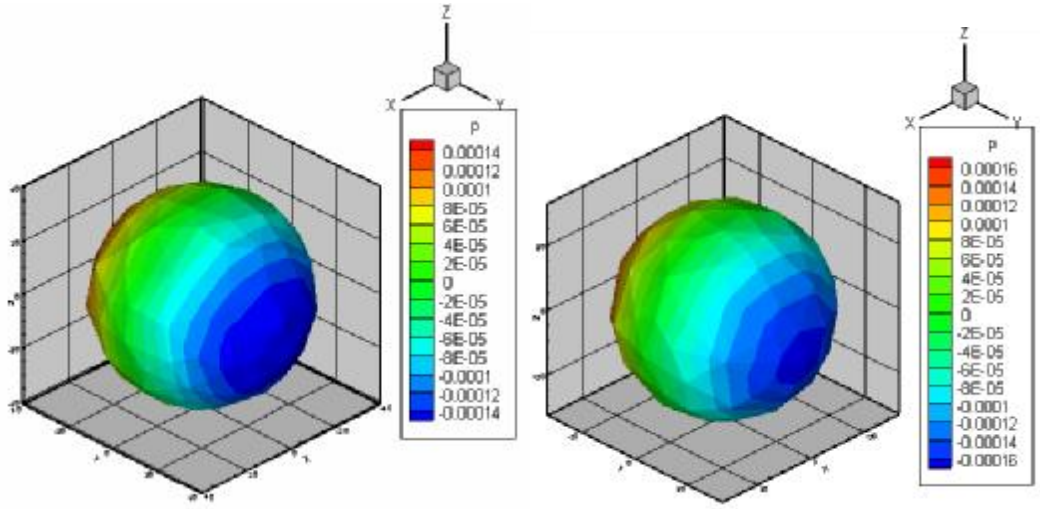


Figure 5-7 Theoretical potential plot on S1 (left) and S4 (right)

5.8.2 Forward solution of the spherical model

The forward solution was used to compute the potentials on surface S1 using the potential on the surface S4. In this test case, the potential on S4 was given by the theoretical calculation. The potential on S1 was assumed unknown and solved by the BEM.

A contour plot of the BEM computation result is given in Figure 5-7. The theoretical calculation and BEM solution are plotted according to element numbers in Figure 5-8.

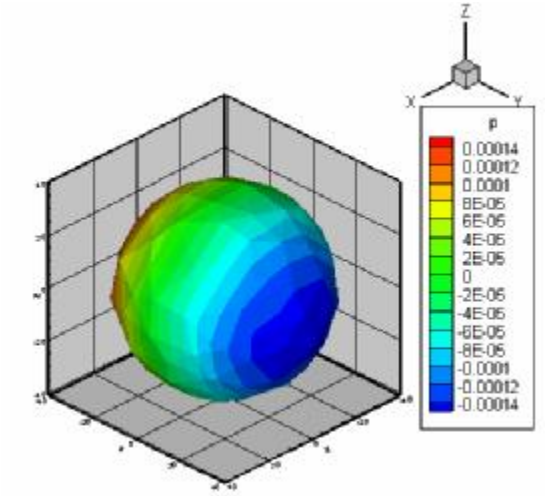
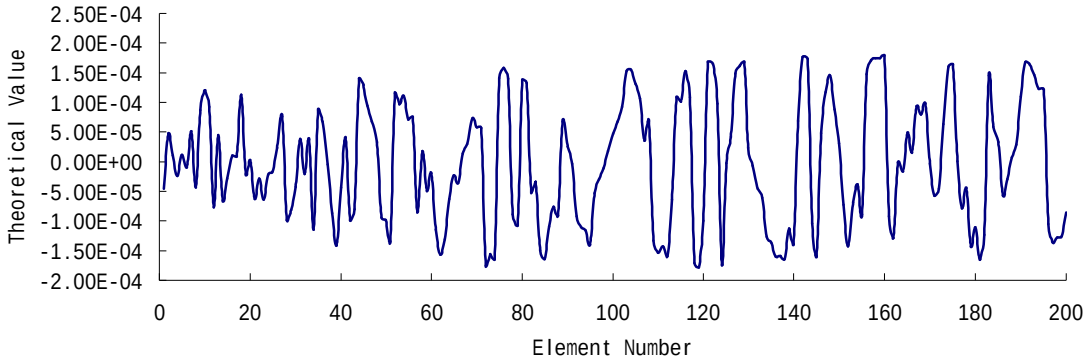
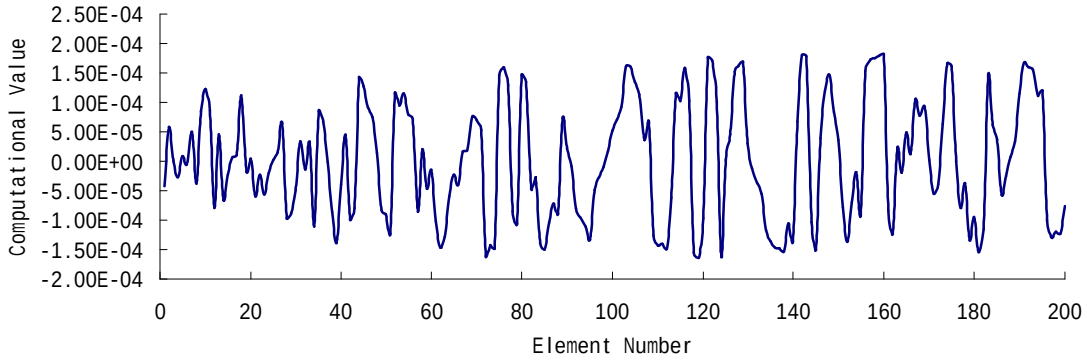


Figure 5-8 Computational potential on S1 using BEM



(a) Theoretical potentials according element number



(b) Computational potentials according element number

Figure 5-9 Potential value according the element number

However, comparisons using contour and curve plots are not sufficient to make a quantitative judgment. Thus we adopt Relative Difference Measures (RDMs) and scatter plots to compare the computation error.

RDM is defined as:

$$RDM = \sqrt{\frac{\sum_{i=1}^{N}(P_i^t - P_i^c)^2}{\sum_{i=1}^{N}(P_i^t)^2}}$$

where P_i^t and P_i^c are the theoretical and computational potentials of the i-th element, N is the total number of boundary elements on the surface.

Scatter plots can visualize the correlation between two variables X and Y (e.g., theoretical and computational values). Individual data points are represented in two-dimensional plot, where axes represent the variables (X on the horizontal axis and Y on the vertical axis).

In the BEM solution of EEG forward problem, RDM=6.68%. A scatter plot is given below, where the points lay close to the line X=Y .

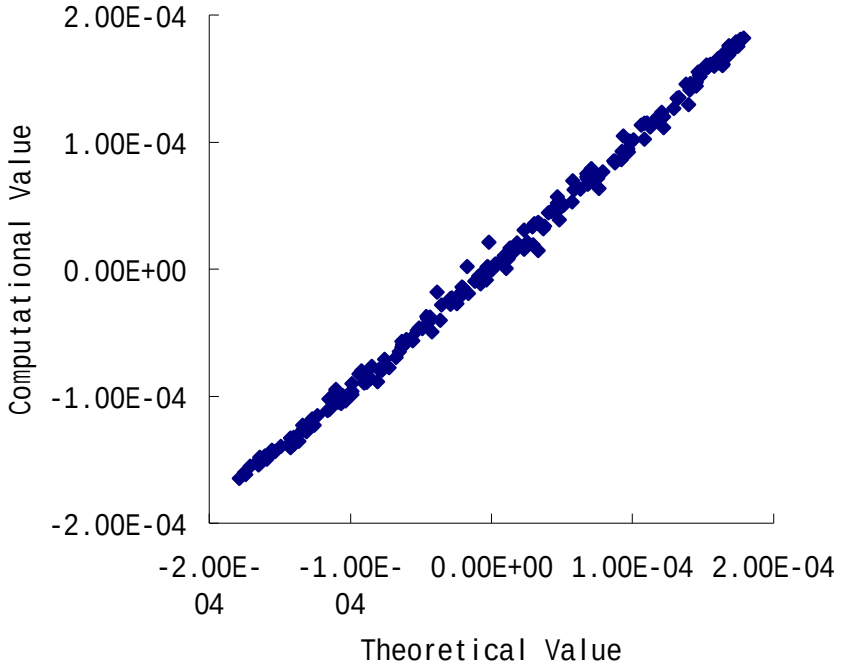


Figure 5-10 Scatter plot of theoretical potential and computational potential in an EEG forward solution

5.8.3 Inverse solution of the spherical model

Using the same mesh in the forward problem, the EEG inverse solution for the potential on S4 was computed using the potential on the surface S1. The potential on S1 was given by the theoretical calculation in section 5.8.1. A contour plot by BEM computation is given below.

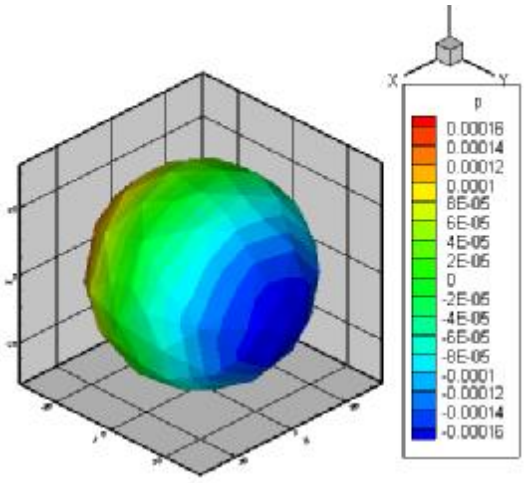


Figure 5-11 Computational potential on S4 using BEM

In this test case, the RDM is 15.3%. And a scatter plot below implies that the inverse solution contains a slight numerical error.

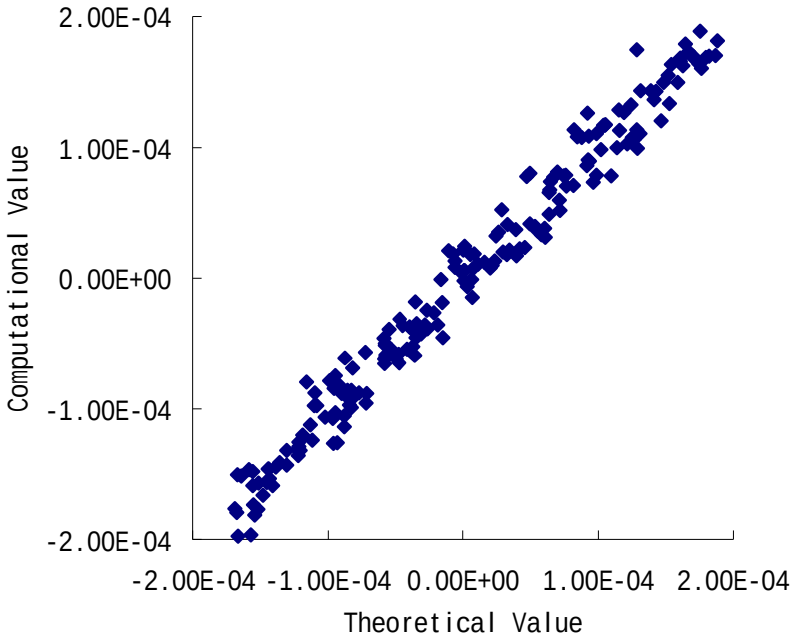


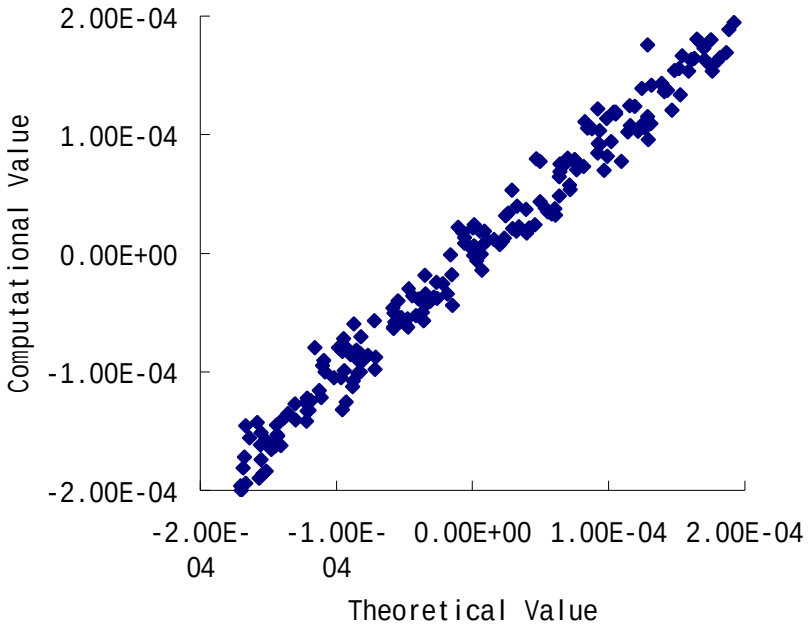
Figure 5-12 Computational potential on S4 using BEM

5.8.4 Influence of white noise signals

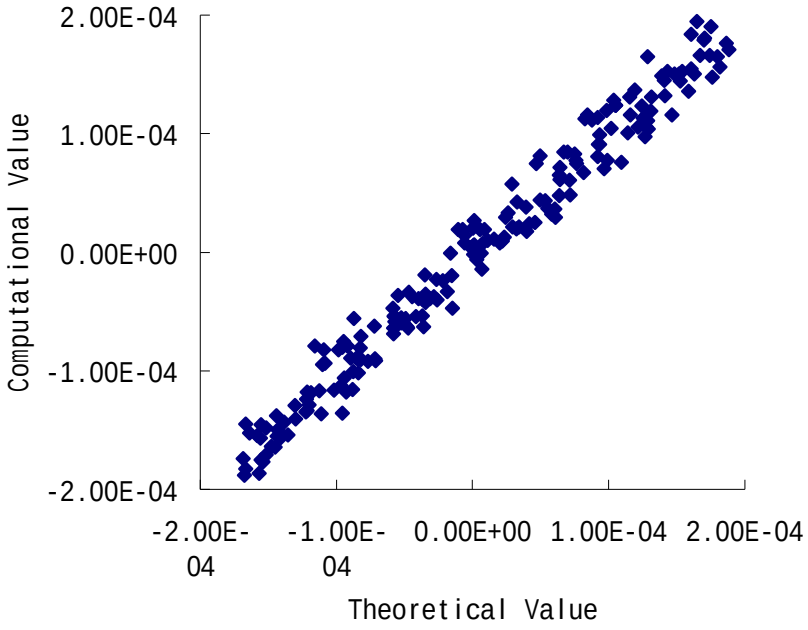
To investigate the influence of noise in measurement, white noise is considered in this section. White noise is a random signal with a flat power spectral density. In other

words, it has equal influence at all signal frequencies. The white noise used in this research is generated by the building function in MATLAB. The noise level is defined as the ratio of standard deviation of noise signal and the root of power of potentials on S1.

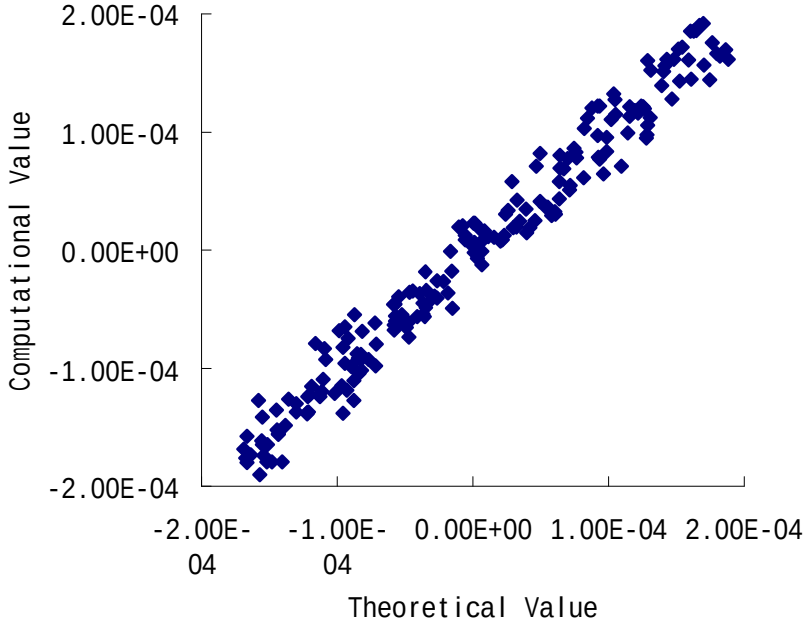
White noise signal of 10%, 20%, and 30% were added to the boundary conditions of EEG inverse solution. Scatter plots are given in Figure 5-13.



(a) Scatter plot of the inverse solution with 10% white noise



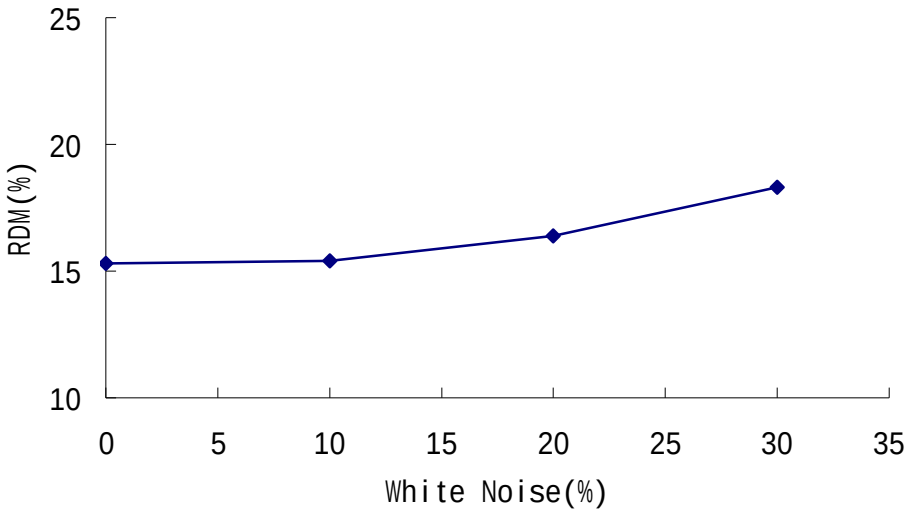
(b) Scatter plot of the inverse solution with 20% white noise



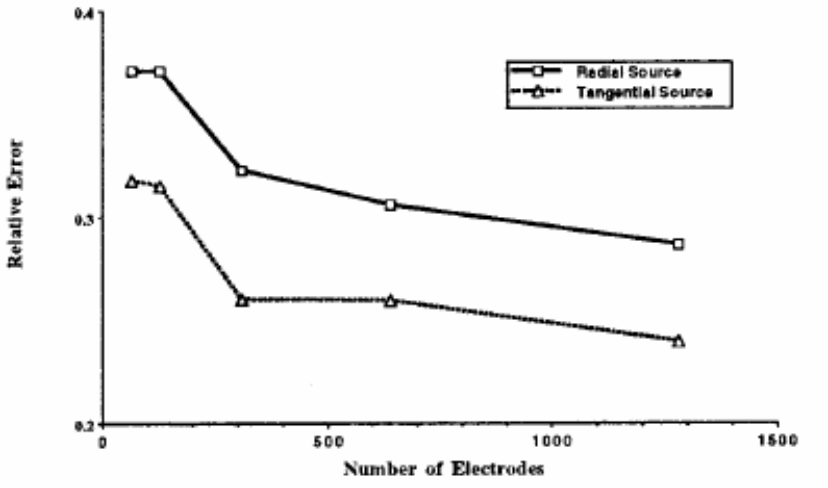
(c) Scatter plot of the inverse solution with 30% white noise

Figure 5-13 Potential value according the element number

By RDM comparison showed in Figure 5-14(a), the numerical error increases with an increasing noise level. Overall, the solution is considered stable even with the noise. Another numerical study reported is shown in Figure 5-14(b). Compared to the reported numerical case [25] with even finer boundary mesh (1280 triangular elements), in which the RDM is about 25% for 10% white noise, the accuracy of the EEG inverse problem is improved in this study.



(a) Effect of white noise on the numerical error in this study



(b) The numerical error reported in previous research [25]

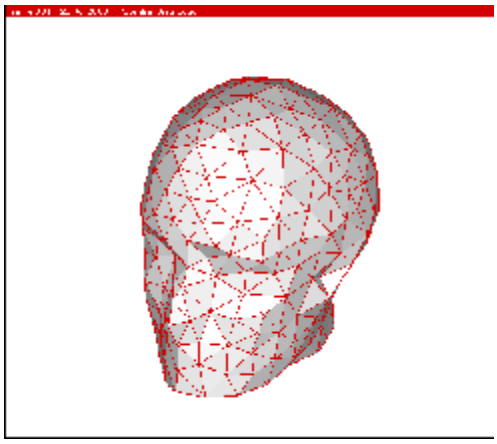
Figure 5-14 Effect of white noise on the numerical error (evaluated by RDM)

5.9 Simulations on the realistic models

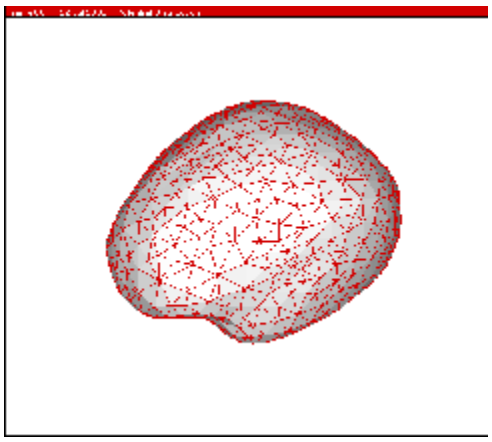
The practical use of the inverse solution of EEG is to compute the cortical potentials from the known scalp potentials from EEG measurements. Realistic models are normally much more complex than the spherical models used for the former tests, because the irregular geometry of human organs contains more details, which need more elements. Thus the block matrix computation and parallel computing become necessary for these large computational cases.

5.9.1 Inverse solution on small-scale models

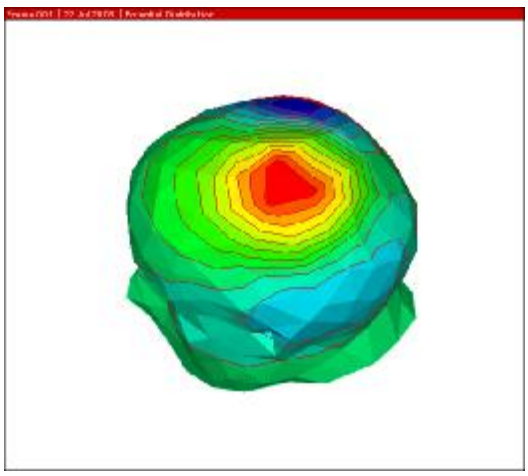
The inverse solution was used to compute the cortical potential by using a given potential on the scalp. First, a relatively simple model is used as the computation case. The brain surface contains 1000 triangular elements while each of other surfaces contains 500 elements. The scalp mesh, which also contains the given potential distribution, and the brain mesh are given by Figure 5-15 (a) and (b). The entire computational process is about 150 minutes. The contour plot is shown in Figure 5-15 (c) and (d).



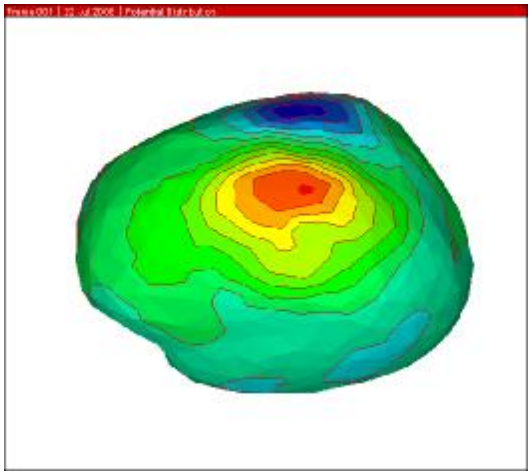
(a) Mesh on the scalp surface



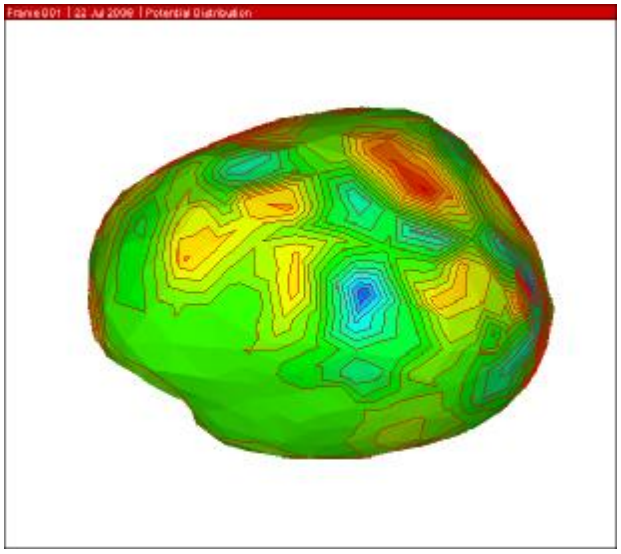
(b) Mesh on the brain surface



(c) Potential given on scalp surface



(d) BEM results by multi- T-SVD

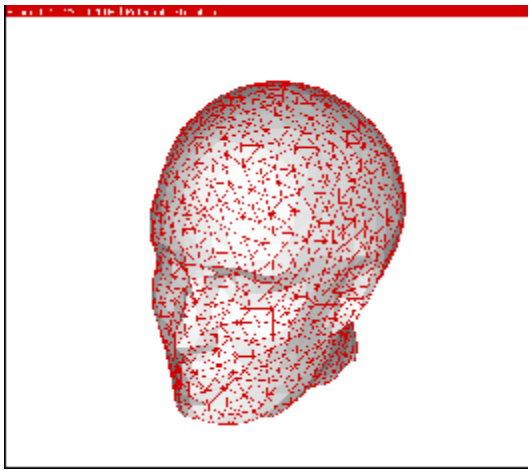


(e) BEM results by single T-SVD

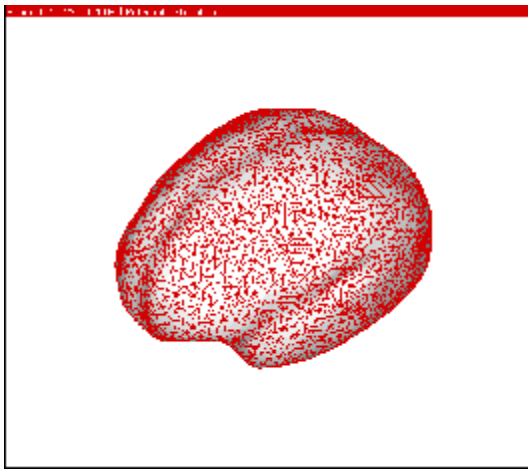
Figure 5-15 EEG inverse solution on realistic model

Some researchers have only applied the truncated SVD once to solve the last pseudoinverse, while other transfer matrices were solved by normal inverse matrix computations [25]. We also conducted a test of this method on the same realistic model. The contour plot is shown in Figure 5-15(e). However, even given the same geometry, triangular meshing and boundary conditions, the inverse solution presented an irregular pattern.

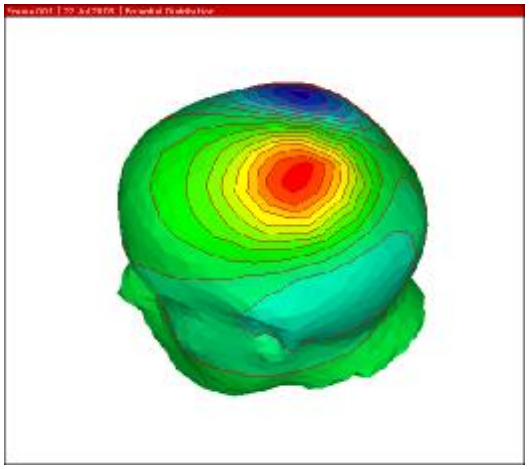
Then a set of fine mesh were used for further study. The brain surface contains 6000 triangular elements while each of other surfaces contains 2000 elements. The scalp mesh, which also contains the given potential distribution, and the brain mesh are given by Figure 5-16 (a) and (b). The entire computational process is about 150 minutes. The contour plot is shown in Figure 5-16 (c). This cortical potential contour plot presents a clearer resolution than that in Figure 5-15 (d).



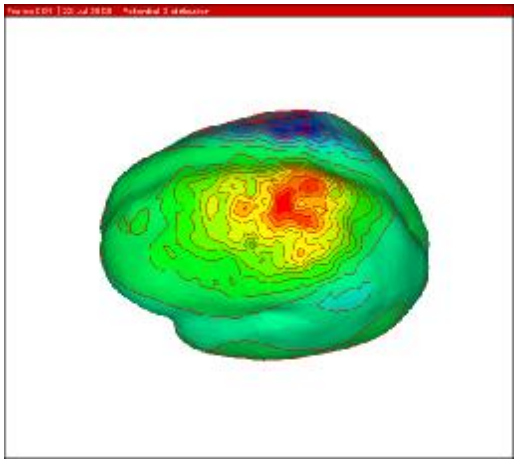
(a) Mesh on the scalp surface



(b) Mesh on the brain surface



(c) Potential given on scalp surface

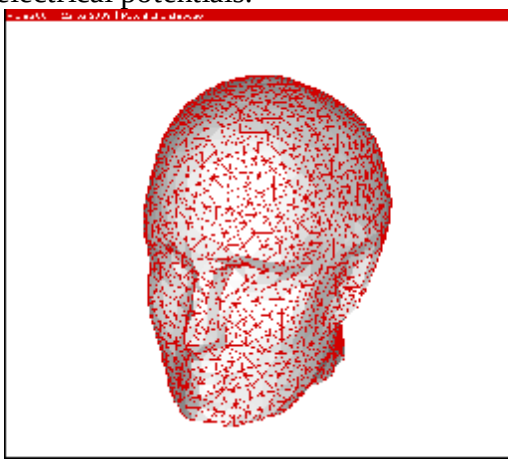


(d) BEM results by multi- T-SVD Figure 5-16 EEG inverse solution on a larger realistic model

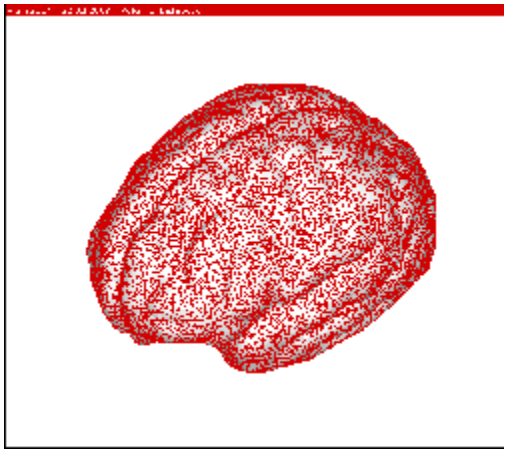
5.9.2 Large-scale simulations on the realistic models

In this section, parallel computing and block matrix computing are used to speed up BEM computation on a much larger model to test the efficiency of this EEG inverse solution. In these large realistic models, the brain surface is discretized into 16,000 elements; the other surfaces are discretized into 3,000 triangular elements each.

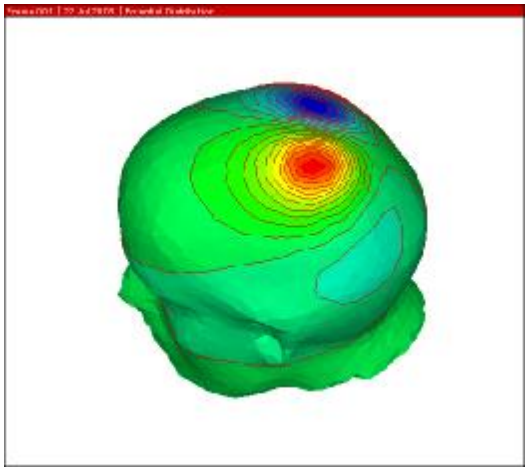
The scalp mesh, which also contain the given potential, and the brain mesh are given by Figure 5-16 (a) and (b), and the computational results are shown in Figure 5-16 (c) and (d). Models with a large number of elements conveyed more details on geometry and electrical potentials.



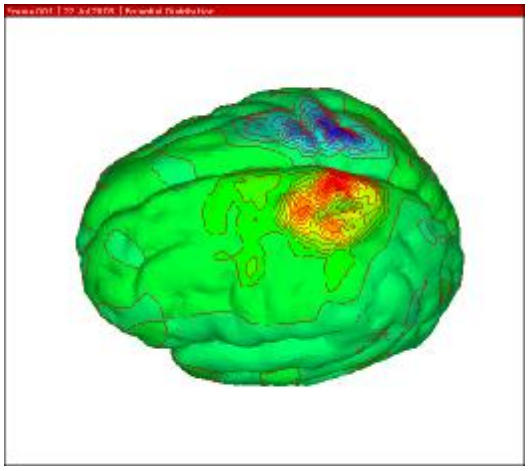
(a) Mesh on the scalp surface



(b) Mesh on the brain surface



(c) Potential given on scalp surface



(d) BEM results by multi- T-SVD Figure

Figure 5-17 Large scale computation of EEG inverse problem on realistic model

5.9.3 Effects of parallel computing and block matrix computing

The block matrix computing method was used in the computation of the transfer matrix between surface S3 and S4. It reduced the memory usage from 6 Gb to 1.5 Gb, which is acceptable for currently using computers.

Parallel computing was employed to compute the transfer matrices between surface S1, S2, and S3. Twelve desktop computers connected in a LAN were used in this process. The practical computation time (wall clock time) was reduced to 2.1 hours from a total computation time of 32.4 hours. The memory usage for each computer was also reduced from 3.3 Gb to 0.3 Gb. The effect of parallel computing is illustrated as below.

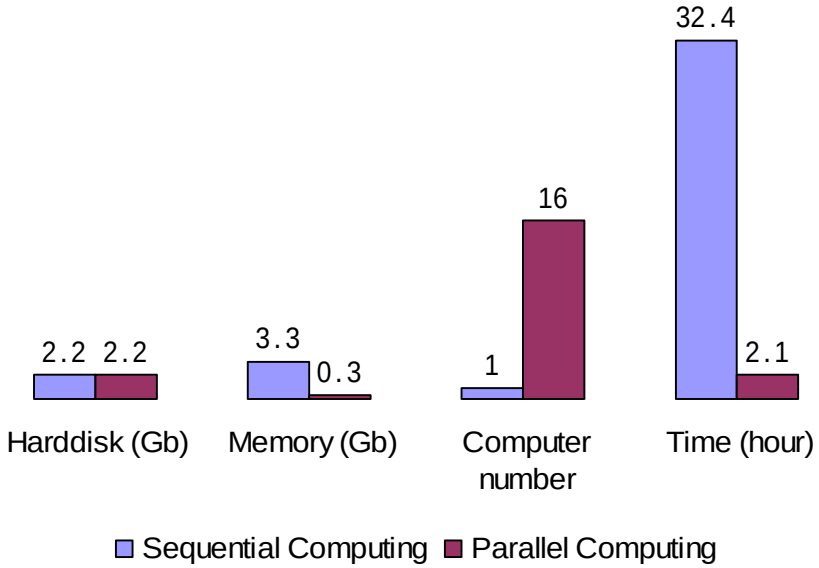


Figure 5-18 Comparison between single serial computing and parallel computing

CHAPTER 6

DISCUSSION AND CONCLUSIONS

6.1 Discussion

As discussed in Chapter 3, the BEM has advantages in the modeling process, and the procedure from later research illustrated this point. The image-based BEM simulation uses the geometric information stored in STL format as a triangular surface mesh directly for computation. FEM simulations have to generate patches, grids, and NURBS surfaces based on the STL surfaces and then build a volume in the FEM software (ANSYS) before meshing it using three dimensional elements. Note that this comparison was made only in the computation step. Since the BEM mesh doesn't require NURBS and solid reconstruction steps in the FEM, time can also be saved in the post-processing step.

Image-based BEM showed an extra advantage in the EEG inverse solution. EEG has a good resolution in time domain, because it may measure as much as a frequency of 100/sec. This means that thousands of boundary condition sets will be computed after the measurement.

For FEM simulation, the time-consuming iteration process is carried out after the potential measurements are input as boundary conditions. For BEM, the time-consuming iteration process, which is to compute the transfer matrices, is carried out before the potential measurements are input as boundary conditions. Thus BEM doesn't need to repeat iteration steps for a same geometry. For hundreds of EEG measurement on a same patient, BEM should accelerate inverse solutions.

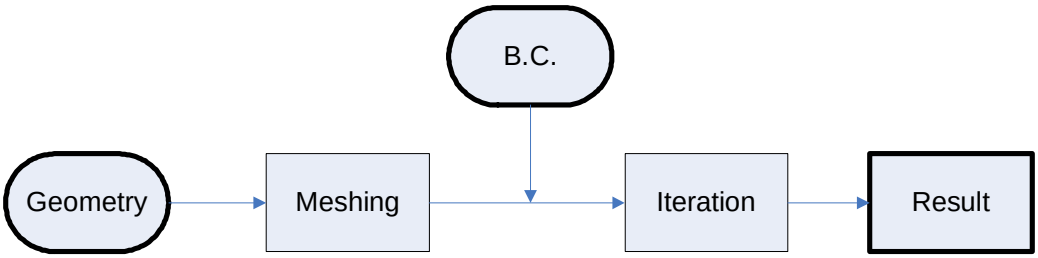


Figure 6-1 Workflow of EEG inverse solution by FEM

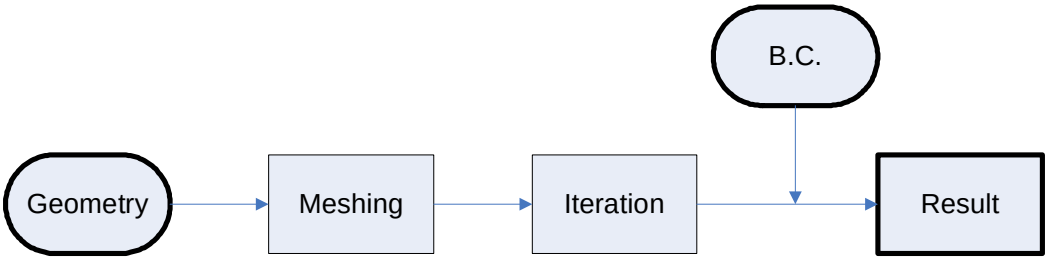


Figure 6-2 Workflow of EEG inverse solution by BEM

6.2 Conclusions

In this research, an integrated image-based boundary element method is developed for engineering simulation of complex freeform objects. This method allows for direct data import of digital scan images for boundary element computation, and therefore it provides advantages over the existing FEM simulation methods, which face time-consuming solid model reconstruction and discretization. A mesh regularization procedure was implemented to improve computation accuracy and speed. Parallel computing and block matrix computing were applied to speed up the conventional BEM computation. Numerical comparisons were conducted on thermal potential and bio-electrical potential problems between the BEM concept and the FEM concept. The efficiency and accuracy of image-based BEM were also demonstrated. Results show the feasibility to apply image-based BEM for digital model simulations.

CHAPTER 7

FUTURE WORK

To further this research work, a stress analysis study on the bone tissue could be conducted. The cross-linking of collagen fibrils stiffens the bone structure, and changes in this cross-linking with age are believed to be the reason why bones more brittle. The micro-scanning and image-based BEM would provide useful information for bone fracture prevention and healing.

More research should also be conducted on EEG electrodes. When EEG measurement is performed, electrodes are located on the scalp, and the spatial resolution of EEG heavily relies on the distribution of electrodes. Numerical study on the distribution could help with clinical operation.

The inverse solution of EEG can be similarly applied to electrocardiogram (ECG) inverse problems. In ECG, electrodes are placed on the skin surface and the electrical activity of the heart is recorded over time. ECG and EEG both compute the electric fields generated by bioelectric sources under quasi-static conditions, and the potential distributions are solutions to the Laplace equation. Thus it should be possible to apply the image-based BEM onto ECG research.

Besides truncated SVD, there are other methods, such as virtual triangle refinement, which can further improve the BEM accuracy in EEG reverse problem. These methods could be adopted in future research.

Lastly, in this research, the optimal choice of a truncation level for the truncated SVD method is determined by quick comparison between numerical experiments. Methods that can automatically determine the truncation level for a certain matrix could be implemented as further improvements.

BIBLIOGRAPHY

[1] A. Werner, K. Skalski, S. Piszczatowski, W. Swieszkowski, Z. Lechniak, "Reverse engineering of free-form surfaces", *Journal of Materials Processing Technology*, vol.76, no.1-3,128-132, 1998.

[2] Remondino Fabio, "From point cloud to surface: the modeling and visualization problem" , *International Archives of the Photogrammetry*, Remote Sensing and Spatial Information Sciences, vol. XXXIV-5, no.10, 2003.

[3] Tamas Varady, Ralph R Martin, Jordan Cox, "Reverse engineering of geometric models--an introduction", *Computer-Aided Design*, vol. 29, no. 4, pp. 255-268, April 1997.

[4] M. Argento, S. Barone, F. Bianconi; P. Conti, E. Rosati,. "Titolo Reverse engineering and CFD analysis: a case study", *International Conference on Applied Simulation and Modelling*, Greece, 2004.

[5] Karbacher, S.; Laboureux, X.; Schon, N.; Hausler, G., "Processing range data for reverse engineering and virtual reality", *Third International Conference on 3-D Digital Imaging and Modeling*, pp.314-321, 2001.

[6] Sabry, El-Hakim, Emily Whiting, Lorenzo Gonzo, Stefano Girardi, "3-D reconstruction of complex architechtures from multiple data", *3D Virtual Reconstruction and Visualization of Comple*x, Italy, 2005.

[7] E. Bassoli, A. Gatto, L. Iuliano, F. Leali, "Design for manufacturing of an ergonomic joystick handgrip", *World Automation Congress*, vol.18, pp.461-466, 2004.

[8] P. A. Webb, "A review of rapid prototyping (RP) techniques in the medical and biomedical sector", *Journal of Medical Engineering & Technology*, vol.24, no.4, 149 – 153, 2000.

[9] S. Singare, L. Dichen, L. Bingheng, L. Yanpu, G. Zhenyu, L. Yaxiong, "Design and fabrication of custom mandible titanium tray based on rapid prototyping", *Medical Engineering & Physics*, vol.26, no.8, pp. 671-676, 2004.

[10] R.M. Gulrajani, "The forward and inverse problems of electrocardiography," *Engineering in Medicine and Biology Magazine, IEEE,* vol.17, no.5, pp.84-101, 122, 1998.

[11] Masashi ENDO, "Reverse Engineering and CAE", *JSME International Journal Series C*, vol. 48, no.2, 218-223, 2005.

[12] S. J. Hollister, B. A. Riemer, “Digital image based finite element analysis for bone microstructure using conjugate gradient and Gaussian filter techniques”, *The International Society for Optical Engineering,* San Diego, pp. 95-106, 1993.

[13] S. A. Langer, E. R Fuller, W.C. Carter, “An image-based finite element analysis of material microstructures”, *Computing in science and engineering*, vol. 3, no.3, pp. 15-23, 2001.

[14] G. T. Charras, R.E.Guldberg, “Improving the local solution accuracy of large-scale digital image-based finite element analyses”, *Journal of Biomechanics*, vol. 33 no.2, pp. 255-259,2000.

[15] T. Kujime, M. Tane, S.K. Hyun, H. Nakajima, “Three-dimensional image-based modeling of lotus-type porous carbon steel and simulation of its mechanical behavior by finite element method”, *Materials Science and Engineering,* A 460-461, pp. 220-226, 2007.

[16] D. E. Anderson, J. R. Cotton, “Mechanical analysis of percutaneous sacroplasty using CT image based finite element models”, *Medical Engineering & Physics* , vol.29, pp. 316-325, 2007.

[17] S. J. Hollister, N. Kikuchi, “Homogenization theory and digital imaging: a basis for studying the mechanics and design principles of bone tissue”, *Biotechnology and Bioengineering*, vol. 43, pp. 586-596, 1994.

[18] B. V. Rietbergen, R. Huiskes, F. Eckstein, P. Rüegsegger, “Trabecular bone tissue strains in the healthy and osteoporotic human femur”, *Journal of Bone and Mineral Research*, vol.18, no.10, pp. 1781-1788, 2003.

[19] M. Fuchs, R. Drenckhahn, H. Wischmann, M. Wagner, "An improved boundary element method for realistic volume-conductor modeling," *IEEE Transactions on Biomedical Engineering*, vol.45, no.8, pp.980-997, Aug 1998.

[20] Jiansheng Yuan; Zhanghong Tang, "Finite-element simulation of human brain electric activity," *IEEE Transactions on Magnetics*, vol.39, no.3, pp. 1539-1542, May 2003.

[21] P.K. Banerjee, “Boundary Element Methods in Engineering”, McGraw-Hill, London, New York, 1994.

[22] Gene H. Golub, Charles F. Van Loan, “Matrix Computations”, Science Press, Beijing, China, 2005.

[23] Xiaolin Chen, Hui Zhang, " An Integrated Imaging and BEM for Fast Simulation of Freeform Objects, " *Computer-Aided Design and Applications*, vol. 4, no.1-4, pp. 371-380, 2008.

[24] Dezhong Yao, "Electric potential produced by a dipole in a homogeneous conducting sphere," IEEE Transactions on Biomedical Engineering, vol.47, no.7, pp.964-966, Jul 2000.

[25] Bin He; Yunhua Wang; Dongsheng Wu, "Estimating cortical potentials from scalp EEGs in a realistically shaped inhomogeneous head model by means of the boundary element method ," *IEEE Transactions on Biomedical Engineering*, vol.46, no.10, pp.1264-1268, Oct 1999.

[26] Manfred Fuchs, Jorn Kastner, Michael Wagner, Susan Hawes and John S. Ebersole, " A standardized boundary element method volume conductor model, " *Clinical Neurophysiology*, vol. 113, no.5, pp. 702-712, May 2002.

APPENDIX

In this research, MATLAB code was developed for both thermal potential study and bio-potential study. The appendix section contains main functions and main subroutines for each numerical study cases to explain the general workflow. In addition, important sub-functions are given to show the computation details. Repeated code and some minor sub-functions are not included.

Appendix A1: Mesh regularization for 3D thermal BEM computation

In the thermal potential study, the code attached in following pages is programmed in MATLAB. Several items listed below could help to understand the general ideas and considerations in the code.

- ThermalPotentialComputation.m is the main function in this thermal potential study. Running of this code will lead automatically to the simulation results. In this file, the first few lines will read .STL files and transfer the data to matrix format. Thus users need to input the full name of their .STL file. In this code, EM and NM are set as global variables, which are necessary to other .m code, such as checkQ.m.

- SubDerV.m is to check if edges in every triangles in the .STL are given by the clockwise sequence, because it is related to the normal direction determination.

- subSTL2M.m is to read .STL and write to tempVM file and tempEM file.

- subT2E.m generate EM and NM based on the previous data. Because the STL file doesn't contain sequential number of elements and nodes.
- subbq.m is to calculate the quality factor of mesh and let the user have a general idea of the mesh.

- subRepairAcuteElement.m and subRepairObtuseElement.m refine the mesh by collapsing and swapping, respectively.

- In the subdatthermal.m file, the thermal boundary conditions can be given according geometry of the model.

- In the subdat.m file, the stress boundary conditions can be given according geometry of the model.

- subnewNM.m rearrange the EM and NM since collapsing operations changed the connectivity by deleting nodes.

- subplt.m gnenerates .plt files for tecplot, which can show the geometry can boundary condition visually.

- subBCplt.m generates the .plt of mesh and boundary condition for tecplot. For later BEM, the input file of thermal problem uses 1 to note a fixed temperature and 2 to note a constant heat flux.

- Finally the BEM solver is copied to current folder and executed. Although running BEM under Matlab may decrease the efficiency a little, the status of BEM computation can be monitored and recorded more easily.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File name: ThermalPotentialComputation.m
% Author: Hui Zhang, ENCS, WSUV. 2007
% Purpose: ThermalPotentialComputation.m is the main function in this
thermal potential study. Running of this code will lead automatically
to the simulation results.
% Inputs: Full name of the .STL file; max iteration times of the mesh
regularization
% Outputs: Regularized mesh for BEM and .plt files for tecplot
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Input the full name of a .STL file here, the .STL file must locate in
the same folder of these .m files

clc;clear all;
fname='bone4n.stl'

%%%%%%%%%%%%%%%%%%%%%%

% Read .STL files and transfer the data to matrix format.
% Subroutine subSTL2M.m is to read .STL and write onto hard disk
% Subroutine subT2E.m generates EM and NM based on the previous data.
% Subroutine subbq.m is to calculate the quality factor of mesh and let
the user have a general idea of the mesh, this subroutine is optional.

subSTL2M(fname);
subT2E;
save space1;
clear all;
global EM NM TM VM; % set global variables.
load space1
subbq

%%%%%%%%%%%%%%%%%%%%%%

% Iteration process of mesh regularization
% Subroutines subRepairObtuseElement.m and subRepairAcuteElement.m
refine the mesh by collapsing and swapping, respectively.

clear all;
load space1;
n=10; % n is the max iteration times of the mesh refining
for k=1:n
    k
    subRepairObtuseElement
    subRepairAcuteElement
end
subRepairObtuseElement
save space2
save space3

%%%%%%%%%%%%%%%%%%%%%%

% Mesh collapsing may remove certain nodes from the node list. Here
'subnewNM' is to update the node matrix
```

```
subnewNM
save space3b

%%%%%%%%%%%%%%%%%%%%%%

% Output B.C. and mesh in .plt files

Subplt % This subroutine generates .plt files of mesh
Subdatthermal % This subroutine defines B.C. for BEM simulation
subBCplt % This subroutine generates .plt files of B.C.

%%%%%%%%%%%%%%%%%%%%%%

% Copy the BEM solver from the root folder to current folder and
execute

copyfile('C:\Program Files\MATLAB71\work\BEM\3D
Thermal\3D_Potential_FMBEM.exe','3D_Potential_FMBEM.exe');
copyfile('C:\Program Files\MATLAB71\work\BEM\3D
Thermal\input.cnd','input.cnd');
!3D_Potential_FMBEM.exe
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File name: subRepairAcuteElement.m
% Author: Hui Zhang, ENCS, WSUV. 2007
% Purpose: To refine the mesh by collapsing
% Inputs: Read .mat data on hard disk under the same folder
% Outputs: Regularized mesh for BEM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%

% Initialize control factor of q

zbad=-65
zratio=0.06%0.15

%%%%%%%%%%%%%%%%%%%%%%

% Show the worst elements before mesh regularization
% Subroutine subCheckQ calculated quality factor of every element

Q=subCheckQ;
subplot(2,2,1), hist(Q)
wq=find(Q==min(Q));
wt=NM((EM(wq(1),:)),:);
wt=[wt;wt(1,:)];
subplot(2,2,2), plot3(wt(:,1),wt(:,2),wt(:,3));
axis equal;


%%%%%%%%%%%%%%%%%%%%%%
% Mark elements worse than the given control factor

bq=find(Q<zbad);
bq(:,2:4)=EM(bq,:);
z=NM(bq(:,2),:)-NM(bq(:,3),:);     %z is a temp variable
bq(:,5)=(sum(z.^2,2));
z=NM(bq(:,4),:)-NM(bq(:,3),:);
bq(:,6)=(sum(z.^2,2));
z=NM(bq(:,2),:)-NM(bq(:,4),:);
bq(:,7)=(sum(z.^2,2));


% bq is a matrix, the first index is element number, the second index
is following information for each element:(number node1 node2 node3
length1^2 length2^2 length3^2,cosine of obtuse angle, Q, small length
ratio)
% Here bq(:,5:7) are the square of length

bq(:,8)=(sum((bq(:,5:7)),2)-2*(max(bq(:,5:7),[],2))) .*
(max(bq(:,5:7),[],2) ./ (4*prod(bq(:,5:7),2))).^0.5;
bq(:,9)=Q(bq(:,1));
bq(:,10)=min(bq(:,5:7),[],2)./sum((bq(:,5:7)),2);


%%%%%%%%%%%%%%%%%%%%%%

% Build a list of elements to collapse
```

```
bq=sortrows(bq,8);
z2=find(bq(:,10)>zratio);
z2s=find(bq(:,10)<=zratio);

%%%%%%%%%%%%%%%%%%%%%%%
% repair acute element
% zalert is used to avoid operating an element pair twice

EMl=size(EM);
EMl=EMl(1);
zalert=zeros(EMl,1);
zdel=[];


for k1=z2s'

    %     to find the neighbor Ele
    ll=find(bq(k1,:)==min(bq(k1,5:7))); %side with shortest length
    if ll==5
        lsn=bq(k1,[2,3]); %nodes# of the shortest side lsn=[node# node#]
    elseif ll==6
        lsn=bq(k1,[3,4]);
    elseif ll==7
        lsn=bq(k1,[2,4]);
    else
        'error'
    end

    %avoid record the data twice
    lzdel=size(zdel);
    lzdel=lzdel(1);
    z4=0;
    for k3=1:lzdel

        if ~isempty(find(zdel(k3,:)==lsn(1))) &
~isempty(find(zdel(k3,:)==lsn(2)))
            z4=1;
        end
    end
    if z4==1
        continue
    end

    % record the nodes pair and element pair to delete
    zdel=[zdel;lsn];


end

EM2=EM;

% NM2=NM;
NMl=size(NM);
NMl=NMl(1);
newNl=1:NMl;
newNl2=newNl;
```

```
% Change old node# to new node#

k5=size(zdel);
k5=k5(1);
for k6=1:k5
    newNl(zdel(k6,2))=newNl(zdel(k6,1));
end

%delete nodes

k8=find(newNl2~=newNl);

for k7=k8
    Nk7=find(EM2==k7);
    EM2(Nk7)=newNl(k7);
end


%delete element
EMdel=[];
for k8=1:EMl
    if EM2(k8,1)==EM2(k8,2)|EM2(k8,1)==EM2(k8,3)|EM2(k8,2)==EM2(k8,3)
        EMdel=[EMdel,k8];
    end
end
EM2(EMdel,:)=[];
EM=EM2;

%%%%%%%%%%%%%%%%%%%%%%

% Show the worst element after repairment

Q2=subCheckQ2;
bq(:,8)=(sum((bq(:,5:7)),2)-2*(max(bq(:,5:7),[],2))) .*
(max(bq(:,5:7),[],2) ./ (4*prod(bq(:,5:7),2))).^0.5;

subplot(2,2,3), hist(Q2)
wq2=find(Q2==min(Q2));
wt2=NM((EM(wq2(1),:)),:);
wt2=[wt2;wt2(1,:)];
subplot(2,2,4), plot3(wt2(:,1),wt2(:,2),wt2(:,3));
axis equal
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File name: subRepairObtuseElement.m
% Author: Hui Zhang, ENCS, WSUV. 2007
% Purpose: To refine the mesh by collapsing
% Inputs: Read .mat data on hard disk under the same folder
% Outputs: Regularized mesh for BEM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%

% Initialize control factor of q
zbad=-65

%%%%%%%%%%%%%%%%%%%%%%%

% Show the worst elements before mesh regularization
% Subroutine subCheckQ calculated quality factor of every element
Q=subCheckQ;
subplot(2,2,1), hist(Q)

wq=find(Q==min(Q));
wt=NM((EM(wq(1),:)),:);
wt=[wt;wt(1,:)];
subplot(2,2,2), plot3(wt(:,1),wt(:,2),wt(:,3));
axis equal;

%%%%%%%%%%%%%%%%%%%%%%%

% Mark elements worse than the given control factor

bq=find(Q<zbad);    % set the criteria for 'bad element'
bq(:,2:4)=EM(bq,:);
z=NM(bq(:,2),:)-NM(bq(:,3),:);     %z is just a temp
bq(:,5)=(sum(z.^2,2));
z=NM(bq(:,4),:)-NM(bq(:,3),:);
bq(:,6)=(sum(z.^2,2));
z=NM(bq(:,2),:)-NM(bq(:,4),:);
bq(:,7)=(sum(z.^2,2));

% bq is a matrix, the first index is element number, the second index
is following information for each element:(number node1 node2 node3
length1^2 length2^2 length3^2,cosine of obtuse angle, Q, small length
ratio)
% Here bq(:,5:7) are the square of length

bq(:,8)=(sum((bq(:,5:7)),2)-2*(max(bq(:,5:7),[],2))) .*
(max(bq(:,5:7),[],2) ./ (4*prod(bq(:,5:7),2))).^0.5;
bq(:,9)=Q(bq(:,1));
bq(:,10)=min(bq(:,5:7),[],2)./sum((bq(:,5:7)),2);

%%%%%%%%%%%%%%%%%%%%%%%

% Build a list of elements to collapse
% obtuse elements will be repaired by ascending sequence of obtuse
cosine

bq=sortrows(bq,8);
```

```
% make the 9th colomn as a sign to show whether the elements are
   already modified once
z2=find(bq(:,10)>0.03);
z2s=find(bq(:,10)<=0.03);

%%%%%%%%%%%%%%%%%%%%%%
% repair obtuse element
% zalert is used to avoid operating an element pair twice

EMl=size(EM);
EMl=EMl(1);
zalert=zeros(EMl,1);

for k1=z2'

%     jishu=k1;
    ll=find(bq(k1,:)==max(bq(k1,5:7))); %side with longest length
    if ll==5
        lsn=bq(k1,[2,3]); %nodes# of the longest side lsn=[node# node#]
    elseif ll==6
        lsn=bq(k1,[3,4]);
    elseif ll==7
        lsn=bq(k1,[2,4]);
    else
        'error'
        ll
        bq(k1,:)
    end

    % find the other element 'opposideE' with the given side

    EMl=size(EM);
    EMl=EMl(1);
    opposideE=[];
    for k2=1:EMl
        if ~isempty(find(EM(k2,:)==lsn(1)))&~isempty( find(EM(k2,:) ==
lsn(2) ))&(k2~=bq(k1,1))
            opposideE=[opposideE;k2];

        end
    end
    z3=size(opposideE);
    if z3(1)~=1
        'error'
        EM(opposideE,:)
        lsn
        continue
    end

    %to avoid reoperate

    if zalert(opposideE)==1&zalert(bq(k1,1))==1
        'these neighbor obtuse elements are already changed'
        continue
    end
```

```
    %find out the end points

    bqend=EM(bq(k1,1),6-find(EM(bq(k1,1),:)==lsn(1))-
find(EM(bq(k1,1),:)==lsn(2)));
    opend=EM(opposideE,6-find(EM(opposideE,:)==lsn(1))-
find(EM(opposideE,:)==lsn(2)));

    %make sure that the opposite Ele is not a thin Ele, if necessary,
      don't operate this element

    lratiao=max([sum((NM(opend,:)-NM(lsn(1),:)).^2),sum((NM(opend,:)-
NM(lsn(2),:)).^2)])/bq(k1,ll);
    if lratiao>25
        'error'
    end

    %generate new element

    z4=find(EM(bq(k1,1),:)==lsn(2)); %lsn(2) will be replaced by opend
    z5=find(EM(opposideE,:)==lsn(1)); %lsn(1) will be replaced by bqend
    EM(bq(k1,1),z4)=opend;
    EM(opposideE,z5)=bqend;

    % new bq

    z6=k1;
    z7=find(opposideE==bq(:,1));
    z6=[z6;z7];

    bq(z6,2:4)=EM(bq(z6),:);
    z=NM(bq(z6,2),:)-NM(bq(z6,3),:);
    bq(z6,5)=(sum(z.^2,2));
    z=NM(bq(z6,4),:)-NM(bq(z6,3),:);
    bq(z6,6)=(sum(z.^2,2));
    z=NM(bq(z6,2),:)-NM(bq(z6,4),:);
    bq(z6,7)=(sum(z.^2,2));

    % make a record to avoid reoperate

    zalert([opposideE,bq(k1,1)])=1;

end

%%%%%%%%%%%%%%%%%%%%%%%%
% Show the worst element after repairment

Q2=subCheckQ2;
bq(:,8)=(sum((bq(:,5:7)),2)-2*(max(bq(:,5:7),[],2))) .*
(max(bq(:,5:7),[],2) ./ (4*prod(bq(:,5:7),2))).^0.5;

subplot(2,2,3), hist(Q2)
wq2=find(Q2==min(Q2));
wt2=NM((EM(wq2(1),:)),:);
wt2=[wt2;wt2(1,:)];
subplot(2,2,4), plot3(wt2(:,1),wt2(:,2),wt2(:,3));
axis equal
```

Appendix A2: The forward and inverse computations of EEG

For bio-potential problems on spherical models, the EEG forward solution and inverse solution used the 'BEMmainForward.m' and 'BEMmainInverse.m', respectively.

z1importSurfData is to read STL files and transfer STL data into matrix data in Matlab, then stored the data. This step requires STL files and their names as inputs.

z2MatrixBuild is a series of functions, which compute the transfer matrices between surfaces. 12 means the scalp skin volume, which contains S1 and S2; similarly, 23 is for the skull bone volume and 34 is the CS fluid volume. Here S1, S2, S3, and S4 are scalp outer, skull outer, skull inner, and brain surfaces, respectively.

z3theoretical is to calculate the potential distribution generated by dipoles. The position, direction and magnitude parameters of dipoles are required as user inputs.

z4forward is to solve the potential on S1 by potential on S4.

z4svdU3noise is to solve the potential on S4 by potential on S1.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File name: BEMmainForward.m
% Author: Hui Zhang, ENCS, WSUV. 2008
% Purpose: This is the main function of the forward EEG solution.
Running of this code will lead automatically to the simulation results.
% Inputs: .STL files; manual input of dipoles input in z3theoretical.m
% Outputs: Results of EEG on the head surface S1 in .plt files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Read STL and translate to mesh
z1importSurfData;

% Read mesh and calculate the Coefficient Matrix between S1 and S2
z2MatrixBuild12

% Read mesh and calculate the Coefficient Matrix between S2 and S3
z2MatrixBuild23

% Read mesh and calculate the Coefficient Matrix between S3 and S4
z2MatrixBuild34

% save all data to datastep2
load z2s12;
load z2s23;
load z2s34;
load datastep1;
save datastep2;

% Calculate the theoretical potential generated by dipoles
z3theoretical;

% Using the potential on S4 brain, solve for potential on S1 scalp
z4forward;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File name: BEMmainForward.m
% Author: Hui Zhang, ENCS, WSUV. 2008
% Purpose: This is the main function of the inverse EEG solution.
Running of this code will lead automatically to the simulation results.
% Inputs: .STL files; manual input of dipoles input in z3theoretical.m
% Outputs: Results of EEG on the brain surface S4 in .plt files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Read STL and translate to mesh
z1importSurfData;

% Read mesh and calculate the Coefficient Matrix between S1 and S2
z2MatrixBuild12

% Read mesh and calculate the Coefficient Matrix between S2 and S3
z2MatrixBuild23

% Read mesh and calculate the Coefficient Matrix between S3 and S4
z2MatrixBuild34

% save all data to datastep2
load z2s12;
load z2s23;
load z2s34;
load datastep1;
save datastep2;

% Calculate the theoretical potential generated by dipoles
z3theoretical;

% Using the potential on S1 scalp, solve for potential on S4 brain
z4svdU3noise;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File name: z1importSurfData.m
% Author: Hui Zhang, ENCS, WSUV. 2008
% Purpose: This is to read STL and translate to mesh
% Inputs: Full names of .STL files
% Outputs: Matrix data in .mat files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% for S1
clc;clear all;

fname='R1.stl';% the file name
t = cputime;
subSTL2M(fname);
subT2E;
cputime-t

EM1=EM; %Element matrix
NM1=NM; %Node matrix
VM1=VM; %Normal vector matrix
save space1;

%%%%%%%%

clc;clear all;

fname='R2.stl';% the file name
t = cputime;
subSTL2M(fname);
subT2E;
cputime-t

EM2=EM;
NM2=NM;
VM2=VM;
save space2;


%%%%%%%%

clc;clear all;

fname='R3.stl';% the file name
t = cputime;
subSTL2M(fname);
subT2E;
cputime-t

EM3=EM;
NM3=NM;
VM3=VM;
save space3;

%%%%%%%%
clc;clear all;

fname='R4.stl';% the file
```

```
t = cputime;
subSTL2M(fname);
subT2E;
cputime-t

EM4=EM;
NM4=NM;
VM4=VM;
save space4;

% Reload the matrix data and save to datastep1
load space1;
load space2;
load space3;


save datastep1 EM1 EM2 EM3 EM4 NM1 NM2 NM3 NM4 VM1 VM2 VM3 VM4;
clc;clear all;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File name: z2MatrixBuild12.m
% Author: Hui Zhang, ENCS, WSUV. 2008
% Purpose: This is to read mesh and calculate the Coefficient Matrix
between S1 and S2
% Inputs: Read the .mat files from hard disk
% Outputs: Matrix data in .mat files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% load input
load datastep1
t = cputime;

% initialize the matrix and vectors
k=1;% permittivity
n1=length(NM1);
EM2=EM2+n1;
EM2=[EM2(:,2),EM2(:,1),EM2(:,3)]; %change direction
EM=[EM1;EM2];
n=length(EM);
NM=[NM1;NM2];
VM=[VM1;-VM2]; %change direction

%element center position
PV=(NM(EM(:,1),:)+NM(EM(:,2),:)+NM(EM(:,3),:))/3;

% initialize matrix
F=zeros(n);
G=zeros(n);

%eps in this computation process
zz=1e-12;
zz2=zz;

%%%%%%%%%%%%%%%
% Iteration of G and F. formulas refer to the thesis content. If the
point is on the line, add pi; if at one endpoint, add theta; if outside,
add 0; if inside, add 2pi

for z1=1:n

    P=PV(z1,:); %center position

    for z2= 1:n

        zVM=VM(z2,:);
        zNM=NM(EM(z2,:),:);
        Dtag=[0 0 0];
        edgetag=[0 0 0];

        %12
        v1P=P-zNM(1,:);
        v2P=P-zNM(2,:);
        v12=zNM(2,:)-zNM(1,:);
        L=norm(v12);
        r1=norm(v1P);
        r2=norm(v2P);
```

```
z=dot(v1P,zVM);

if z1==z2
    z=0;
end
if abs(z)<zz
    z=0;
end

l1=dot(v1P,v12)/L;
l2=dot(v2P,v12)/L;
vn=cross(zVM,v12)/L;

D=dot(vn,v1P);
if abs(D)<zz2
    Dtag(1)=0;D=0;
    if abs(abs(l1)+abs(l2)-L)<zz
        edgetag(1)=1;
    end
else
    Dtag(1)=sign(D);
end

        zG12=D*log((r1+r2+L)/(r1+r2-L))+z*(-atan(z*l2/(D*r2)) +
         atan(z*l1/(D*r1)));
         zF12=atan(z*l2/(D*r2))-atan(z*l1/(D*r1));

%23
v3P=P-zNM(3,:);
v23=zNM(3,:)-zNM(2,:);
L=norm(v23);
r3=norm(v3P);

l2=dot(v2P,v23)/L;
l3=dot(v3P,v23)/L;
vn=cross(zVM,v23)/L;

D=dot(vn,v2P);
if abs(D)<zz2
    Dtag(2)=0;D=0;
    if abs(abs(l2)+abs(l3)-L)<zz
        edgetag(2)=1;
    end
else
    Dtag(2)=sign(D);
end

         zG23=D*log((r2+r3+L)/(r2+r3-L))+z*(-atan(z*l3/(D*r3)) +
         atan(z*l2/(D*r2)));
          zF23=atan(z*l3/(D*r3))-atan(z*l2/(D*r2));

%31

v31=zNM(1,:)-zNM(3,:);
L=norm(v31);

l3=dot(v3P,v31)/L;
```

```
        l1=dot(v1P,v31)/L;
        vn=cross(zVM,v31)/L;

        D=dot(vn,v3P);

        if abs(D)<zz2
            Dtag(3)=0;D=0;
            if abs(abs(l3)+abs(l1)-L)<zz
                edgetag(3)=1;
            end
        else
            Dtag(3)=sign(D);
        end

        zG31=D*log((r3+r1+L)/(r3+r1-L))+z*(-atan(z*l1/(D*r1)) +
        atan(z*l3/(D*r3)));
        zF31=atan(z*l1/(D*r1))-atan(z*l3/(D*r3));


        alpha=0;
        if sum(Dtag)==3
            alpha=2*pi;
        elseif sum(edgetag)==1
            alpha=pi;
        elseif sum(edgetag)==2
            if edgetag(3)==0
                alpha=acos(-dot(v12,v23)/(norm(v12)*norm(v23)));
            elseif edgetag(1)==0
                alpha=acos(-dot(v23,v31)/(norm(v23)*norm(v31)));
            elseif edgetag(2)==0
                alpha=acos(-dot(v31,v12)/(norm(v31)*norm(v12)));
            end
        end

        F(z1,z2)=dot([zF12 zF23 zF31],abs(Dtag))+sign(z)*alpha;
        G(z1,z2)=dot([zG12 zG23 zG31],abs(Dtag))-abs(z)*alpha;

    end
end

F=F/(4*pi);
G=G/(4*pi*k);

for z1=1:n
    F(z1,z1)=-sum(F(z1,:));
end

zindex1=1:length(EM1);
zindex2=1:length(EM2);
zindex2=zindex2+length(EM1);

Fa11=F(zindex1,zindex1);
Fa22=F(zindex2,zindex2);
Fa12=F(zindex1,zindex2);
Fa21=F(zindex2,zindex1);

Ga11=G(zindex1,zindex1);
```

```
Ga22=G(zindex2,zindex2);
Ga12=G(zindex1,zindex2);
Ga21=G(zindex2,zindex1);

t = cputime-t;

% save data
save z2s12;
save (num2str(t),'t')
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File name: z3theoretical.m
% Author: Hui Zhang, ENCS, WSUV. 2008
% Purpose: This is to read mesh and calculate the Coefficient Matrix
between S1 and S2
% Inputs: Read the .mat files of geometry from hard disk; user input of
dipole parameters
% Outputs: Theoretical potential on all surfaces, stored in .mat files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Formulas of this function refer to [24]
clear all;clc;
load datastep2;

% User inputs of dipoles. Here M2 is much smaller than M1, the pattern
appear to be one dipole. If M2 is of the similar scale of M1, the
pattern are for two diples.
M1=[0,-1,0];
Mr1=[0 0 -2];
M2=[0,1e-6,0];
Mr2=[0 0 2];

dlt=1;
R1=norm(NM1(1,:));
R2=norm(NM2(1,:));
R3=norm(NM3(1,:));
R4=norm(NM4(1,:));
R01=norm(Mr1);
R02=norm(Mr2);
nn1=length(EM1);
nn2=length(EM2);
nn3=length(EM3);
nn4=length(EM4);

%TPL is for the theoretical potential list on specific nodes [1:nn1+nn2]
TPL=zeros(nn1+nn2+nn3+nn4,1);
PV=zeros(nn1+nn2+nn3+nn4,3);
PV(1:nn1,:)=(NM1(EM1(:,1),:)+NM1(EM1(:,2),:)+NM1(EM1(:,3),:))/3;
PV(nn1+1:nn1+nn2,:)=(NM2(EM2(:,1),:)+NM2(EM2(:,2),:)+NM2(EM2(:,3),:))/3;
PV(nn1+nn2+1:nn1+nn2+nn3,:)=(NM3(EM3(:,1),:)+NM3(EM3(:,2),:)+NM3(EM3(:,
3),:))/3;
PV(nn1+nn2+nn3+1:nn1+nn2+nn3+nn4,:)=(NM4(EM4(:,1),:)+NM4(EM4(:,2),:)+NM
4(EM4(:,3),:))/3;

for z=1:nn1 %outer shpere
    zcos1=dot(M1,PV(z,:))/(norm(M1)*norm(PV(z,:)));
    zcos2=dot(M2,PV(z,:))/(norm(M2)*norm(PV(z,:)));
    rp1=norm(PV(z,:)-Mr1);
    rp2=norm(PV(z,:)-Mr2);
    ztemp1=2*(PV(z,:)-Mr1)/rp1^3+(PV(z,:)+(PV(z,:)*R01*zcos1-
R1*Mr1)/(R1+rp1-R01*zcos1))/(R1^2*rp1);
    ztemp2=2*(PV(z,:)-Mr2)/rp2^3+(PV(z,:)+(PV(z,:)*R02*zcos2-
R1*Mr2)/(R1+rp2-R02*zcos2))/(R1^2*rp2);
    %change 1 to 2 for M1 Mr1 R01
    TPL(z)=dot(M1,ztemp1)/(4*pi*dlt)+dot(M2,ztemp2)/(4*pi*dlt);
end
for z=(nn1+1):(nn1+nn2) %inner shpere
```

```
    zcos1=dot(M1,PV(z,:))/(norm(M1)*norm(PV(z,:)));
    zcos2=dot(M2,PV(z,:))/(norm(M2)*norm(PV(z,:)));
    rp1=norm(PV(z,:)-Mr1);
    rp2=norm(PV(z,:)-Mr2);
    rpi1=sqrt(1+(R01*R2/R1^2)^2-2*zcos1*R01*R2/R1^2);
    rpi2=sqrt(1+(R02*R2/R1^2)^2-2*zcos2*R02*R2/R1^2);

    ztemp1=(PV(z,:)-Mr1)/rp1^3  +  (PV(z,:)-(R2/R1)^2*Mr1)/(R1*rpi1)^3
+  (PV(z,:) + (PV(z,:)*R01*R2*zcos1-R2^2*Mr1)/((rpi1+1)*R1^2-
R01*R2*zcos1) )/(R1^3*rpi1);
    ztemp2=(PV(z,:)-Mr2)/rp2^3  +  (PV(z,:)-(R2/R1)^2*Mr2)/(R1*rpi2)^3
+  (PV(z,:) + (PV(z,:)*R02*R2*zcos2-R2^2*Mr2)/((rpi2+1)*R1^2-
R02*R2*zcos2) )/(R1^3*rpi2);


    TPL(z)=dot(M1,ztemp1)/(4*pi*dlt)+dot(M2,ztemp2)/(4*pi*dlt);
end
for z=(nn1+nn2+1):(nn1+nn2+nn3) %inner shpere
    zcos1=dot(M1,PV(z,:))/(norm(M1)*norm(PV(z,:)));
    zcos2=dot(M2,PV(z,:))/(norm(M2)*norm(PV(z,:)));
    rp1=norm(PV(z,:)-Mr1);
    rp2=norm(PV(z,:)-Mr2);
    rpi1=sqrt(1+(R01*R3/R1^2)^2-2*zcos1*R01*R3/R1^2);
    rpi2=sqrt(1+(R02*R3/R1^2)^2-2*zcos2*R02*R3/R1^2);

    ztemp1=(PV(z,:)-Mr1)/rp1^3  +  (PV(z,:)-(R3/R1)^2*Mr1)/(R1*rpi1)^3
+  (PV(z,:) + (PV(z,:)*R01*R3*zcos1-R3^2*Mr1)/((rpi1+1)*R1^2-
R01*R3*zcos1) )/(R1^3*rpi1);
    ztemp2=(PV(z,:)-Mr2)/rp2^3  +  (PV(z,:)-(R3/R1)^2*Mr2)/(R1*rpi2)^3
+  (PV(z,:) + (PV(z,:)*R02*R3*zcos2-R3^2*Mr2)/((rpi2+1)*R1^2-
R02*R3*zcos2) )/(R1^3*rpi2);


    TPL(z)=dot(M1,ztemp1)/(4*pi*dlt)+dot(M2,ztemp2)/(4*pi*dlt);
end

for z=(nn1+nn2+nn3+1):(nn1+nn2+nn3+nn4) %inner shpere
    zcos1=dot(M1,PV(z,:))/(norm(M1)*norm(PV(z,:)));
    zcos2=dot(M2,PV(z,:))/(norm(M2)*norm(PV(z,:)));
    rp1=norm(PV(z,:)-Mr1);
    rp2=norm(PV(z,:)-Mr2);
    rpi1=sqrt(1+(R01*R3/R1^2)^2-2*zcos1*R01*R3/R1^2);
    rpi2=sqrt(1+(R02*R3/R1^2)^2-2*zcos2*R02*R3/R1^2);

    ztemp1=(PV(z,:)-Mr1)/rp1^3  +  (PV(z,:)-(R3/R1)^2*Mr1)/(R1*rpi1)^3
+  (PV(z,:) + (PV(z,:)*R01*R3*zcos1-R3^2*Mr1)/((rpi1+1)*R1^2-
R01*R3*zcos1) )/(R1^3*rpi1);
    ztemp2=(PV(z,:)-Mr2)/rp2^3  +  (PV(z,:)-(R3/R1)^2*Mr2)/(R1*rpi2)^3
+  (PV(z,:) + (PV(z,:)*R02*R3*zcos2-R3^2*Mr2)/((rpi2+1)*R1^2-
R02*R3*zcos2) )/(R1^3*rpi2);


    TPL(z)=dot(M1,ztemp1)/(4*pi*dlt)+dot(M2,ztemp2)/(4*pi*dlt);
end


%TPLN is the potential value on nodes
```

```
TPLN1=zeros(length(NM1),2);
for z1=1:nn1
    TPLN1(EM1(z1,:),1)=TPLN1(EM1(z1,:),1)+TPL(z1);
    TPLN1(EM1(z1,:),2)=TPLN1(EM1(z1,:),2)+1;
end

TPLN2=zeros(length(NM2),2);
for z1=1:nn2
    TPLN2(EM2(z1,:),1)=TPLN2(EM2(z1,:),1)+TPL(nn1+z1);
    TPLN2(EM2(z1,:),2)=TPLN2(EM2(z1,:),2)+1;
end


TPLN3=zeros(length(NM3),2);
for z1=1:nn3
    TPLN3(EM3(z1,:),1)=TPLN3(EM3(z1,:),1)+TPL(nn1+nn2+z1);
    TPLN3(EM3(z1,:),2)=TPLN3(EM3(z1,:),2)+1;
end

TPLN4=zeros(length(NM4),2);
for z1=1:nn4
    TPLN4(EM4(z1,:),1)=TPLN4(EM4(z1,:),1)+TPL(nn1+nn2+nn3+z1);
    TPLN4(EM4(z1,:),2)=TPLN4(EM4(z1,:),2)+1;
end

TPLN1=TPLN1(:,1)./TPLN1(:,2);
TPLN2=TPLN2(:,1)./TPLN2(:,2);
TPLN3=TPLN3(:,1)./TPLN3(:,2);
TPLN4=TPLN4(:,1)./TPLN4(:,2);

% Write the theoretical results in .plt files
subztheoreticalplot;
save datastep3;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File name: z4forward.m
% Author: Hui Zhang, ENCS, WSUV. 2008
% Purpose: This is to read mesh and B.C. and calculate the EEG forward
solution
% Inputs: Read the .mat files of mesh from hard disk; read theoretical
potential on S4 generated by dipoles as a B.C.
% Outputs: Computational potential on S1, plotted by .plt
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% In this computation the matrix T and S are abbreviation form of the
middle step, which is in formula (9), page 39 of the thesis. The names
of matrix are following the procedure in Reference [25]

load datastep3;
delta1=1;
delta2=1;
delta3=1;

% T21 and S21

T21=inv(Ga12*inv(Ga22)*Fa22-Fa12)*(Fa11-Ga12*inv(Ga22)*Fa21);
S21=inv(Ga22)*(Fa21+Fa22*T21);

% T31 and S31

T31=inv(Gb23*inv(Gb33)*Fb33-Fb23)  *  ((Fb22-Gb23*inv(Gb33)*Fb32)*T21 +
(Gb22-Gb23*inv(Gb33)*Gb32)*S21*delta1/delta2);
S31=inv(Gb33)*(Fb32*T21+Fb33*T31+Gb32*S21*delta1/delta2) ;

% A and B

A=((Fc33-Gc34*inv(Gc44)*Fc43)*T31+(Gc33-
Gc34*inv(Gc44)*Gc43)*S31*delta2/delta3);
B=(Gc34*inv(Gc44)*Fc44-Fc34);


% A B ->solution
T=A'*inv(A*A')*B;

U1=T*TPL(nn1+nn2+nn3+1:end);

NM=NM1;
EM=EM1;
n=nn1;

PLN2=zeros(length(NM),2);
for z1=1:n
    PLN2(EM(z1,:),1)=PLN2(EM(z1,:),1)+U1(z1);
    PLN2(EM(z1,:),2)=PLN2(EM(z1,:),2)+1;
end

PLN=PLN2(:,1)./PLN2(:,2);

% write the results to .plt files

fid = fopen('computational potential U1.plt', 'wt');
```

```
fprintf(fid, ' TITLE = "Potential Distribution" \n');
fprintf(fid, 'VARIABLES = "X", "Y", "Z", "p", "m", "n"\n');
fprintf(fid, ' ZONE DATAPACKING=POINT, ZONETYPE=FETRIANGLE,
N=%8d ,E=%8d\n',max(size(NM)),max(size(EM)));


NBM=zeros(size(NM));
NBM=[NBM;NBM];
NBM(1:2:end,:)=NM;
NBM(2:2:end,1)=PLN;

fprintf(fid, '%+13.7E %+13.7E %+13.7E\n',NBM');
fprintf(fid, '\n');

fprintf(fid, ' %14d %14d %14d\n',EM');
fprintf(fid, '\n');
fclose(fid);

plot(1:nn1,TPL(1:nn1,1),'b',1:nn1,U1(1:nn1,1),'r');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File name: z4svdU3noise.m
% Author: Hui Zhang, ENCS, WSUV. 2008
% Purpose: This is to read mesh and B.C. and solve EEG inverse solution
% Inputs: Read the .mat files of mesh from hard disk; read theoretical
potential on S1 generated by dipoles as a B.C.
% Outputs: Computational potential on S4, plotted by .plt
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% In this computation the matrix T and S are abbreviation form of the
middle step, which is in formula (9), page 39 of the thesis. The names
of matrix are following the procedure in Reference [25]

load datastep3;
delta1=1;
delta2=1;
delta3=1;

% T21 and S21

T21=inv(Ga12*inv(Ga22)*Fa22-Fa12)*(Fa11-Ga12*inv(Ga22)*Fa21);
S21=inv(Ga22)*(Fa21+Fa22*T21);

% T31 and S31


    T31=inv(Gb23*inv(Gb33)*Fb33-Fb23)  *  ((Fb22-Gb23*inv(Gb33)*Fb32)*T21 +
                    (Gb22-Gb23*inv(Gb33)*Gb32)*S21*delta1/delta2);
S31=inv(Gb33)*(Fb32*T21+Fb33*T31+Gb32*S21*delta1/delta2) ;

% A and B

A=((Fc33-Gc34*inv(Gc44)*Fc43)*T31+(Gc33-
Gc34*inv(Gc44)*Gc43)*S31*delta2/delta3);
B=(Gc34*inv(Gc44)*Fc44-Fc34);


% A B -> Usvd3, Usvd3 is the potential solution on S4. In this case,
the white noise level is 0.3

T=A'*inv(A*A')*B;
[U,S,V] = svd(T);
zinvS=zeros(size(S'));
for z1=1:117
    zinvS(z1,z1)=1/S(z1,z1);
end
Usvd3=V*zinvS*U'*TPL(1:nn1).*(1+0.3*(rand(size(TPL(1:nn1)))-0.5));


NM=NM4;
EM=EM4;
n=nn4;

PLN2=zeros(length(NM),2);
for z1=1:n
    PLN2(EM(z1,:),1)=PLN2(EM(z1,:),1)+Usvd3(z1);
```

```
    PLN2(EM(z1,:),2)=PLN2(EM(z1,:),2)+1;
end

PLN=PLN2(:,1)./PLN2(:,2);

fid = fopen('computational potential plotSVD3.plt', 'wt');
fprintf(fid, ' TITLE = "Potential Distribution" \n');
fprintf(fid, 'VARIABLES = "X", "Y", "Z", "p", "m", "n"\n');
fprintf(fid, ' ZONE DATAPACKING=POINT, ZONETYPE=FETRIANGLE,
N=%8d ,E=%8d\n',max(size(NM)),max(size(EM)));



NBM=zeros(size(NM));
NBM=[NBM;NBM];
NBM(1:2:end,:)=NM;
NBM(2:2:end,1)=PLN;

fprintf(fid, '%+13.7E %+13.7E %+13.7E\n',NBM');
fprintf(fid, '\n');

fprintf(fid, ' %14d %14d %14d\n',EM');
fprintf(fid, '\n');
fclose(fid);
plot(1:nn4,TPL(nn1+nn2+nn3+1:end,1),'b',1:nn4,Usvd3(1:nn4,1),'r');
```

Appendix A3: The large-scale inverse computation of EEG

For the bio-potential problems on large-scale realistic models, the EEG inverse solution used the 'BEMmain'. Several functions are almost the same as Appendix A2, thus they are not include in this section.

z1importSurfData is the subroutine to read STL files and transfer this data into matrix data in Matlab, then stored the data. Full names of files should be given in this part.

z2theoretical is to calculate the potential distribution generated by dipoles. The direction and magnitude parameters can be set for at most two dipoles. The dipole must be within head surface

z3MatrixBuild123 is to calculate the matrix F and G between S1/S2,S2/S3. This part can be executed on parallel computers.

z4T13 is calculate the transfer matrix T13 by F an G matrices.

z5MatrixBuild34 is to calculate the matrix F and G between S3/S4. This part uses the block matrix commutating. Sub-matrices are stored separately.

z6T14 is to calculate the transfer matrix T14. This part also uses the block matrix commutating.

z7pick1000 is choosing 1000 nodes by a given STL file, of which name must be given in this subroutine. The transfer matrix on selected nodes can mimic the electrodes in EEG. In the thesis, these electrodes are 1,000 elements generated by Geomagic on upper part of the head geometry.

z8svdU3 is solving T41 by T14 using the truncated SVD technique.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File name: BEMmain.m
% Author: Hui Zhang, ENCS, WSUV. 2008
% Purpose: This is the main function in the large-scale EEG inverse
solution. Running of this code will lead automatically to the
simulation results.
% Inputs: .STL files; manual input of dipoles input in z2theoretical.m
% Outputs: Results of EEG on the brain surface in .plt files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;clc;
delta1=1;
delta2=1/80;
delta3=1;
save delta

z1importSurfData; % This step requires inputs of STL files and their
full names

z2theoretical; % This step requires inputs of dipole parameters

z3MatrixBuild123;

z4T13

z5MatrixBuild34

z6T14

z7pick1000 % This step requires inputs of a STL file, which contain the
'Electrodes' information

z8svdU3
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File name: z3MatrixBuild123.m
% Author: Hui Zhang, ENCS, WSUV. 2008
% Purpose: This function is to compute the coefficient matrix F and G
between surfaces S1/S2 and S2/S3.
% Inputs: mesh information from .mat files on the hard disk
% Outputs: matrix data stored in .mat files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear

load delta
save datastep3

% Here are 16 subroutines. They can be executed on 16 computers at the
same time, as parallel computing. 'a','b' note different volumes.
subF(1,1,2,delta1,'a')
subF(1,2,2,delta1,'a')
subF(2,2,2,delta1,'a')
subF(2,1,2,delta1,'a')

subG(1,1,2,delta1,'a')
subG(1,2,2,delta1,'a')
subG(2,2,2,delta1,'a')
subG(2,1,2,delta1,'a')

subF(2,2,3,delta2,'b')
subF(2,3,3,delta2,'b')
subF(3,3,3,delta2,'b')
subF(3,2,3,delta2,'b')

subG(2,2,3,delta2,'b')
subG(2,3,3,delta2,'b')
subG(3,3,3,delta2,'b')
subG(3,2,3,delta2,'b')
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File name: z5MatrixBuild34.m
% Author: Hui Zhang, ENCS, WSUV. 2008
% Purpose: This function is to compute the coefficient matrix F and G
between surfaces S3/S4.
% Inputs: mesh information from .mat files on the hard disk
% Outputs: matrix data stored in .mat files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear

load delta
load datastep1
save datastep6 %notice this is not datastep5


% Here are 8 subroutines. They can be executed on 8 computers at the
same time, as parallel computing.

subF2(3,3,4,delta3,'c')
subF2(3,4,4,delta3,'c')
subF2(4,4,4,delta3,'c')
subF2(4,3,4,delta3,'c')

subG2(3,3,4,delta3,'c')
subG2(3,4,4,delta3,'c')
subG2(4,4,4,delta3,'c')
subG2(4,3,4,delta3,'c')
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File name: z6T14.m
% Author: Hui Zhang, ENCS, WSUV. 2008
% Purpose: This function is to compute the transfer matrix T14 between
surfaces S1/S4 by TSVD.
% Inputs: coefficients information from .mat files on the hard disk
% Outputs: matrix data stored in .mat files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear
save datastep6

% Gc and Fc must be stored in mat file "Fc.mat" and "Gc.mat"
% Here several sub-function are used: subdivide, subinv4, subminus, and
subproduct. They are the functions for block matrix computing, because
the original matrix is too large to load. Subdivide is to divide a
matrix into sub-matrices. Other sub-functions are computing the
subtractions and multiplications of sub-matrix.

subdivide('Gc44',4,4,16,16,'Gc44')%we have Gc44pxpy now!

subGc2Gz%divide Gc44 to 4 submatrix group, file Gc44 can be deleted  Gz
is a temp, Gi is also aa temp as the inv of Gc

subinv4('Gz11','Gzt1','Gz11','Gzt1')

subproduct('Gz21','Gzt1','Gzt2',2,2,2,4e3,4e3)
subproduct('Gzt1','Gz12','Gzt3',2,2,2,4e3,4e3)
subproduct('Gzt2','Gz12','Gzt4',2,2,2,4e3,4e3)
subminus('Gz22','Gzt4','Gzt5',2,2)
subinv4('Gzt5','Gi22','Gzt5','Gi22')
subminusproduct('Gzt3','Gi22','Gi12',2,2,2,4e3,4e3)
subminusproduct('Gi22','Gzt2','Gi21',2,2,2,4e3,4e3)
subproduct('Gi12','Gzt2','Gzt6',2,2,2,4e3,4e3)
subminus('Gzt1','Gzt6','Gi11',2,2)
subGi2iGc

%% A and B.

zhnum=3;
subdivide('Gc34',1,4,zhnum,16,'Gc34')%we have Gc34pxpy now!
subproduct('Gc34','iGc','p0',1,4,4,3e3,4e3)% p0=Gc34*iGc44

subdivide('Fc43',4,1,16,zhnum,'Fc43')%we have Fc43pxpy now!
subproduct('p0','Fc43','p1',1,4,1,3e3,3e3)% p1=p0*Fc43

subdivide('Gc43',4,1,16,zhnum,'Gc43')%we have Gc43pxpy now!
subproduct('p0','Gc43','p2',1,4,1,3e3,3e3)% p2=p0*Gc43

subdivide('Fc44',4,4,16,16,'Fc44')%we have Fc44pxpy now!
subproduct('p0','Fc44','p3',1,4,4,3e3,4e3)% p3=p0*Fc44
```

```
subdivide('Fc34',1,4,zhnum,16,'Fc34')%we have Fc34pxpy now!
subminus('p3','Fc34','B',1,4)% B=(p3-Fc34)


subdivide('Fc33',1,1,zhnum,zhnum,'Fc33')
subdivide('Gc33',1,1,zhnum,zhnum,'Gc33')

clear
load p2
load Gc33
temp1=(Gc33p1p1-p2p1p1);
save datastep6 temp1 -append

clear
load S31
load delta
load datastep6 temp1
temp2=temp1*S31*delta2/delta3;
save datastep6 temp2 -append

clear
load p1
load Fc33
temp3=(Fc33p1p1-p1p1p1);
save datastep6 temp3 -append

clear
load T31
load datastep6 temp3
temp4=temp3*T31;
save datastep6 temp4 -append

clear
load datastep6 temp2 temp4
A=temp2+temp4;
save datastep6 A -append


% T14

clear
load datastep6 A
temp5=A*A';
save datastep6 temp5 -append

clear
load datastep6 temp5
temp6=zinv(temp5,200);
save datastep6 temp6 -append

clear
load datastep6 A
load datastep6 temp6
p4p1p1=A'*temp6;
```

```
save p4 p4p1p1

clear
load datastep6 A
p4p1p1=zinv(A,200);
save p4 p4p1p1
clear

subproduct('p4','B','T14',1,1,4,3e3,4e3)% T14=p4*B;%manual operation to
file
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File name: z7pick1000.m
% Author: Hui Zhang, ENCS, WSUV. 2008
% Purpose: This function is to compute the transfer matrix T14 between
surfaces S1/S4 by TSVD.
% Inputs: coefficients information from .mat files on the hard disk
% Outputs: matrix data stored in .mat files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% this code is to pick 1000 point out from a 20k-DOF mesh
% Use CAD software to generate a 1000-DOF mesh and import as 1k.stl
% 20K-DOF mesh is NM1 EM1

clear
fname='1000.stl';

subSTL2M(fname);
subT2E;

%Now we have NM and EM

%center position
load( 'datastep1.mat','NM1','EM1');
PM1k=(NM(EM(:,1),:)+NM(EM(:,2),:)+NM(EM(:,3),:))/3;
PM1=(NM1(EM1(:,1),:)+NM1(EM1(:,2),:)+NM1(EM1(:,3),:))/3;


z1k=size(PM1k);
list1k=zeros(z1k(1),1);


for z1=1:z1k(1)
    a=meshgrid(PM1k(z1,:),1:3000);
    d=sum((PM1-a).^2,2);
    list1k(z1)=find(d==min(d));%get the closest points
end

%%
list1k=sort(list1k);

%%
save('datastep7','list1k')%list1k

%%
load T14 T14p1p1
T=T14p1p1(list1k,:);
clear T14p1p1

load T14 T14p1p2
T=[T,T14p1p2(list1k,:)];
clear T14p1p2

load T14 T14p1p3
T=[T,T14p1p3(list1k,:)];
```

```
clear T14p1p3

load T14 T14p1p4
T=[T,T14p1p4(list1k,:)];
clear T14p1p4

save('datastep7','T','-append')%T
```