國　立　交　通　大　學

機　械　工　程　學　系

博士論文

使用調適網格加密功能之有限元素平行化三維 Poisson-Boltzmann Equation 程式之發展與驗證

Development and Verification of a 3-D Parallelized Poisson-Boltzmann Equation Solver Using Finite Element Method with Adaptive Mesh Refinement

研 究 生：邵雲龍

指導教授：吳宗信 博士

2006 年 07 月

使用調適網格加密功能之有限元素平行化三維 Poisson-Boltzmann Equation 程式之發展與驗證

Development and Verification of a 3-D Parallelized Poisson-Boltzmann Equation Solver Using Finite Element Method with Adaptive Mesh Refinement

研 究 生：邵雲龍　　Student：Yun-Long Shao

指導教授：吳宗信 博士　　Advisor：Dr. Jong-Shinn Wu

國 立 交 通 大 學

機 械 工 程 學 系

博 士 論 文

A Thesis

Submitted to Institute of Mechanical Engineering Collage of Engineering

National Chiao Tung University

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Mechanical Engineering

July 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年七月

致謝

首先，感謝吳宗信教授這幾年的悉心教誨與熱心的指導，讓我在課業上或生活上均受益良多，在此表達由衷的感謝與敬意。且感謝各位口試委員：洪哲文教授、傅武雄教授、陳維新教授、江仲驊教授與黃楓南教授不吝指正，提出許多的寶貴意見，使得本論文的內容更加完備。

同時要感謝 MuST 所有的成員，包括國賢在理論與數值模擬上的大力協助、又永熱心協調平行計算設備資源的分配、允民在計算分析與欣芸在有限元素法技巧上的幫忙、捷粲在平行化程式上的協助、凱文在網格繪製上的幫助，以及坤樟、祐霖、哲維、富利熱情的鼓勵。此外，也要特別感謝已經畢業的碩士班學弟妹們的鼓勵與支持，讓我有源源不斷地動力繼續完成我的學業。

最後，要感謝國家高速網路與計算中心提供叢集式電腦資源、國家奈米元件實驗室與交大奈米中心提供我學習相關設備的機會。

邵雲龍　謹誌

2006 年 07 月 風城交大

Development and Verification of a 3-D Parallelized Poisson-Boltzmann Equation Solver Using Finite Element Method with Adaptive Mesh Refinement

Student：Yun-Long Shao Advisors：Dr. Jong-Shinn Wu

Department of Mechanical Engineering
National Chiao Tung University

Abstract

Understanding of the interactive particle-particle and particle-wall forces in colloidal systems (electrolytes) plays a very important role in bio-chemistry related research. By assuming equilibrium in the electrolytes, the potential distribution with a very thin electrical double layer near the charged object can be well described by the well-known Poission-Boltzmann equation. Thus, a parallelized 3-D nonlinear Poisson-Boltzmann equation solver (PPBES) using finite element method (FEM) with parallel adaptive mesh refinement (PAMR) is proposed and verified. In this thesis, the research is divided into two phases, which are described as follows.

In the first phase, the nonlinear Poisson-Boltzmann equation is discretized using Galerkin finite element method with unstructured tetrahedral mesh. Interpolation within a typical element includes the first-order and second-order shape functions. Inexact Newton iterative scheme is used to solve the nonlinear matrix equation resulting from the FE discretization. Jacobian matrix resulting from the Newton iterative scheme is diagonalized using nodal quadrature, which further facilitates the

easier parallel implementation. A parallel conjugate gradient (CG) method with a subdomain-by-subdomain (SBS) scheme is then used to solve the linear algebraic equation each iterative step. Completed code is verified using two typical examples. The first validated case is the potential distribution around a charged sphere. Excellent agreement of the simulation results with analytical (linearized case) and approximate (nonlinear case) solutions are obtained. The second validated case is the interaction between like-charged spheres within a cylindrical pore. Results show that the agreement between the present simulation and previous results are excellent. The above two typical simulations validate the present implementation of the PPBES. Further, the parallel performance is studied on a HP PC-cluster system at NCHC using a test case with a charged sphere confined in a cylindrical pore. Results show that 76.2% of parallel efficiency can be reached at processors of 32. Also FE discretization using the second-order shape function is demonstrated to be more accurate than using the first-order shape function at the end of this phase.

In the second phase, an *h*-refinement based PAMR scheme for an unstructured tetrahedral mesh using dynamic domain decomposition on a memory-distributed machine is developed and tested in detail. A memory-saving cell-based data structure is designed such that the resulting mesh information can be readily utilized in both node- or cell-based numerical methods. The general procedures include isotropic refinement

from one parent cell into eight child cells and then followed by anisotropic refinement, which effectively removes the hanging nodes, with a simple mesh-quality control scheme. Parallel performance of this PAMR is studied on a PC-cluster system up to 64 processors. Results show that the parallel speedup scales approximately as $N^{1.5}$ up to 32 processors, where N is the number of processors. Then, procedure of coupling the PPBES with the PAMR using *a posteriori* error estimator is presented and verified using a test case with two like-charged spheres in a cylindrical pore. Results show that PAMR can systematically increase the solution accuracy of the PPBES. Finally, the coupled PPBES-PAMR code is used to simulate the interactive force between two like-charged spheres near a charged planar wall. Results are in excellent agreement with experimental data considering the experimental uncertainties, which is the first simulation in the literature to the best knowledge of the author.

Keywords: parallel Poisson-Boltzmann equation solver (PPBES), finite element method (FEM), parallel adaptive mesh refinement (PAMR), *a posteriori* error estimator.

使用調適網格加密功能之有限元素平行化三維 Poisson-Boltzmann Equation 程式之發展與驗證

學生：邵雲龍　　指導教授：吳宗信 博士

國立交通大學機械工程學系博士班

中文摘要

探討微粒-微粒與微粒-平板之間的相互作用力在生物化學的領域是相當重要的。在假設電解液為平衡的狀態下，可以經由著名的 Poisson-Boltzmann 方程式電雙層理論得到帶電物體的勢能分佈。因此，我們決定開發以有限元素法三維平行化非線性 Poisson-Boltzmann 方程式程式(PPBS)搭配平行化調適網格加加密功能程式(PAMR)，並且完成驗證的工作。本論文之研究分成兩大主軸，分別敘述如下：

第一部分，以非線性 Poisson-Boltzmann 方程式採用非結構性四面體網格之葛勒金有限元素法來完成程式開發，包括了典型的一階與二階的形狀函數元素。因為使用了 nodal quadrature 的技巧，使得原先的牛頓法 Jacobian 矩陣僅剩下對角線，以此擬牛頓疊代法來處理非線性項矩陣的部份，有助於平行化程式的完成。接下來，則使用 SBS 的技巧搭配平行的共軛梯度法來處理線性矩陣方程組。完成的程式以兩個範例來驗證，第一個驗證的範例是帶電球體的勢能分佈，所得答案與解析解、近似解作一比較，結果相當正確。第二個驗證範例則是兩個帶電球體在圓柱孔內，結果顯示與先前所發表的論文結果相符。以上兩個驗證範例證明了平行化 Poisson-Boltzmann 方程式程式的開發完成。此外，平行化效能使用

國家高速網路與計算中心的 HP 叢集式電腦系統來做驗證，測試一個帶電球體在圓柱孔內的範例，結果顯示使用了 32 顆 CPU 時仍有 76.2%的平行效能。在第一部份的最後，我們使用了二階形狀函數元素，所得結果證明了比一階形狀函數元素要來的準確。

第二部份，主要是發展一個以非結構性四面體網格為主，採用 *h*-切割為基礎之分散式記憶體動態領域分解的 PAMR 程式，而資料結構則使用了較節省記憶體空間之 cell-base 方式來記錄網格的資訊，可以同時運用在 node-base 與 cell-base 的數值方法上。一般的步驟包括了一個分為八個網格的等向性切割，然後以非等向切割搭配網格品質控制將 hanging node 有效率地移除。我們測試 PAMR 在叢集式電腦上最多 64 顆 CPU 的平行化效能，結果呈現在 32 顆 CPU 時仍有 $N^{1.5}$ 的效能(N 為 CPU 個數)。接著我們將 PPBES 與 PAMR 做一個結合，並使用 *a posteriori* 的誤差評估方法，驗證兩個帶電球體在圓柱孔內，結果證明了使用 PAMR 可以增加 PPBES 解的準確度。最後，利用 PPBES-PAMR 的程式模擬兩個帶電球體靠近帶電平板的相互影響力之分析，與實驗結果相比較，若考慮實驗本身的不確定因素與誤差，模擬結果與實驗結果有相當程度的符合，這是目前已知與實驗結果相比較之最佳模擬結果。

關鍵字: 平行Poisson-Boltzmann方程式求解系統、有限元素法、 平行搭配調適網格功能、*a posteriori*誤差評估方法

Table of Contents

List of Tables

List of Figures

Symbols

a	The radius of the particle (m)
E	The electric field vector
e	The charge of a proton (C)
F	The force acting on the spherical particle (Newton)
h	The distance between plate and particle (m)
N_i	FEM shape function
n_i	Concentration of type i ion (m^{-3})
n_{i0}	Bulk ionic concentration of type i ion (m^{-3})
r	The separation distance of particles (m)
T	Absolute temperature (K)
z_i	Valence of type i ion

Greek symbols

ψ	Electrical potential (V), and $\overline{\psi} = ze\psi / k_b T$
λ	A cylindrical pore with the sphere radius to pore radius ratio
λ_d	Debye length (m)
ρ_e	The net charge density (C/m^3)
ε	Dielectric constant

ε_0	Permittivity of vacuum (C/mV)
κ	Debye-Huckel parameter (m^{-1})
κ_b	Boltzmann constant (J/K)
δ	The global absolute error
δ_m	The current mean absolute error
δ_{pre}	The prescribed global relative error
$\Delta\Pi$	The osmotic pressure

Chapter 1 Introduction

1-1 Motivation

Understanding of the electrostatic distribution in the electrical double layer (EDL) near charged particles or charged solid wall is crucial in understanding several important physical problems, including microscopic colloidal system [Okubo and Aotani,1998; Quddus and Moussa, 2004], electrokinetics in microfluidic system [Quddus and Moussa, 2004; Shao *et. al.*, 2006; Collins and Lee, 2004], to name a few. Behavior of the EDL in a solution can be approximated by the nonlinear Poisson-Boltzmann equation (PBE). The Poisson-Boltzmann equation has been traditionally used to find the electrostatic potential around a macromolecule. Numerical investigation of models based on the PB equation can provide important information on effective particle interaction in colloidal systems.

The Poisson-Boltzmann equation is a second-order partial differential equation with a hyperbolic sine term that makes it exponentially nonlinear in the electrical double layer around the charged objects. Analytical solution is only possible for some problems with simple geometries and boundary conditions. However, most of the applications are three-dimensional with very complicated geometries, which necessitate the efficient and accurate numerical simulation of the Poisson-Boltzmann equation. Thus, in this thesis we intend to present and validate a parallelized

three-dimensional Poisson-Boltzmann equation solver (PPBES) using the finite-element method (FEM), coupling with parallel adaptive mesh refinement (PAMR), which can be used efficiently and accurately for practical 3-D applications in the future.

1-2 Background

In this section, several important concepts about the microscopic colloidal behavior that is relevant to the Poisson-Boltzmann equation are described first. Then adaptive mesh refinement which is important to the accuracy and efficiency in solving the Poisson-Boltzmann equation is described in turn.

1-2-1 Poisson-Boltzmann Equation

1-2-1-1 Nature of Colloidal Solutions

Microscopic particles of one phase dispersed in another are generally called colloidal solution or dispersions. Three fundamental forces operate on fine particles in solution:

1. gravitational force, depending on particles density relative to the solvent;
2. viscous drag force, often described by Stoke's Law, which depicts the force as a function of fluid viscosity, particle diameter, and particle settling

velocity;

3. the ‘natural’ kinetic energy of particles and molecules.

The ‘natural’ kinetic energy including electrostatic force, which is an important factor of interaction for particles and molecules.

Fig. 1-1 shows the typical electrostatic potential distribution around a positively charged colloidal particle. Negative counter-ions are attracted towards the particle by the electric field generated by the positively charged surface, but they are also subject to thermal motion, which tends to spread then uniformly through the surrounding medium.

1-2-1-2 Zeta Potential

Almost all particulate or macroscopic materials in contact with a liquid acquire an electronic charge on their surfaces. Zeta potential is an important and useful indicator of this charge which can be used to predict and control the stability of colloidal suspensions or emulsions.

Colloidal particles dispersed in a solution are electrically charged due to their ionic characteristics and dipolar attributes. Each particle dispersed in a solution is surrounded by oppositely charged ions called the fixed layer shown as Fig. 1-2. Outside the fixed layer, there are varying compositions of ions of opposite polarities,

forming a cloud-like area. This area is called the diffuse double layer, and the whole area is electrically neutral in general.

When a voltage is applied to the solution in which particles are dispersed, particles are attracted to the electrode of the opposite polarity, accompanied by the fixed layer and part of the diffuse double layer, or internal side of the "sliding surface".

Zeta potential is considered to be the electric potential of this inner area including this conceptual "sliding surface". As this electric potential approaches zero, particles tend to aggregate.

1-2-1-3 Electric Double Layer

Electrokinetic phenomena are of considerable importance in many fields of science and engineering. In particular, they exert a strong influence on the flow behavior of a fluid, for example, in microchannels and capillaries.

The earliest studies of the electric double layer near a metal surface, by Helmholtz [Hunter, 1989] in the mid-nineteenth century, assumed that the electric charge on the metal surface was balanced by an equal charge which sat in the surrounding solutions a little way from the surface. Around 1910 a better model was proposed by a French scientist name Gouy and a similar treatment was developed

independently a few years later by the British scientist, Chapman. The result is known as the Gouy-Chapman model [Hunter, 1989] of the electric double layer, modified by Helmholtz model. It assumes that most solid surfaces bear electrostatic charges. When a charge surface is in contact with an electrolyte, the electrostatic charges on the solid surface will influence the distribution of nearby ions in the electrolyte solution. Ions of opposite of charges to that of the surface are attracted towards the surface, while ions of like charges are repelled from the surface; thus, an electric field is established. The charges on the solid surface and the balancing charge in the liquid are called the "electric double layer" (EDL). The EDL potential distribution is show in Fig 1-3. The sign and magnitude of the EDL field depend on the nature of the surface and the liquid.

Immediately next to the charged solid surface, there is a layer of ions that are strongly attracted to the solid surface and are immobile. This layer is called the compact or stern layer, normally about several Angstroms thick. The charge and potential distributions in this layer are mainly determined by the geometrical restriction of ion molecule size and the short-range interactions between ions, the wall and the adjoining dipoles.

From the compact layer to the electrically neutral bulk liquid, the net charge density gradually reduces to zero. Ions in this region are affected less strongly by the electrostatic interaction and are mobile. This region is called the diffuse layer of the

EDL. As will be discussed later, an equation called the Poisson-Boltzmann equation is used to describe approximately the ion and potential distributions in the diffuse layer.

The EDL width, called Debye length and afterward noted, determines the electric interaction range between macromolecules and therefore controls the static phase behavior of these systems.

1-2-1-4 Applications of Poisson-Boltzmann Equation

The Poisson-Boltzmann equation (PBE) provides a "fast" and approximate method for calculating the effects of solvent on electrostatic interactions in the modeled system. There are at least five major applications of the PBE:

1. Calculation of the electrostatic potential at the surface of a biomolecule, which is expected to give information about the concentration of small charged solutes in the neighborhood of the molecule and whose inspection may suggest docking sites for biomolecules [Bowen and Sharif, 1998];
2. Calculation of the electrostatic potential outside the molecule, which is expected to give information on the free energy of interaction of small molecules at different positions in the surrounding of the molecule. [Kozack and Subramaniam, 1993];
3. Calculation of the free energy of a biomolecule or of different states of a

biomolecule which gives information on the stability of a biomolecule or of its different states [Sharp and Honig, 1990];

4. Calculation of the electrostatic field to derive mean forces to be added in standard molecular dynamics calculations [Gilson *et. al*., 1993].
5. Calculation of the electrostatic force among charged particles/walls, which is important in the study of colloidal system.

In the present thesis, we are mainly concerned about the last application of the Poisson-Boltzmann equation, to which it is not limited. The related derivation of Poisson-Boltzmann Equation will be discussed in chapter 2.

1-2-2 Adaptive Mesh Refinement (AMR)

Adaptive mesh refinement (AMR) is a computational technique for improving the efficiency and accuracy of solution of numerical simulations. In the AMR technique we start with a base coarse grid. As the solution proceeds we identify the regions requiring more resolution by some parameter characterizing the solution. We superimpose finer subgrids only on these regions. Finer and finer subgrids are added recursively until either a given maximum level of refinement is reached or the local truncation error has dropped below the desired level.

In general, mesh refinement can be categorized into three methods: (1) local

polynomial-degree-variation (*p*-refinement*)*; (2) mesh movement (*r*-refinement); and (3) mesh enrichment (*h*-refinement), which will be briefly described respectively in the following.

1-2-2-1 Local Polynomial-Degree-Variation (*p*-refinement*)*

As shown in Fig. 1-4, local *p*-refinement scheme increases resolution of the solution by increasing the order of polynomial of the shape function in the element, which originated from finite element modeling. However, its implementation in three-dimensional finite-element modeling is rather complicated and not as popular as the mesh refinement (*h*-refinement) which will be introduced later.

1-2-2-2 Mesh Movement (*r*-refinement)

As shown in Fig. 1-5, the total mesh points remain the same in the computational domain after refinement. It is common to use a spring analogy, in which the nodes of the mesh are connected by springs whose stiffness is proportional to certain measure of solution activity over the spring. The mesh points are moved closer into the region where solution gradients are relatively large. This is often applied to the spatial adaptation of a structured mesh.

1-2-2-3 Mesh Enrichment (*h*-refinement)

As shown in Fig 1-6, with mesh enrichment, mesh points are added or embedded into the regions where relatively large solution gradients are detected, while the global mesh topology remains intact. It is generally regarded that mesh enrichment method has certain advantages over the first two methods. For example, the use of the *h*-refinement scheme generally does not impact the coding structure of the original numerical scheme if hanging nodes are properly treated, while the use of *p*-refinement does, thus complicating the practical implementation. One of the most important advantages is that the mesh enrichment technique is in general many times faster and robust than the remeshing technique.

1-3 Literature Surveys

1-3-1 Numerical Simulation of Poisson-Boltzmann Equation

The Poisson-Boltzmann equation plays an important role in describing many physical phenomena, including like-charge particle interaction [Dyshlovenko, 2001; Squires and Brenner, 2000; Tuinier, 2003], particle-membrane interaction [Bowen, 1998; Tuinier, 2003], and electrokinetic flow [Yang *et. al*., 2001], among others. For example, one of the applications, colloidal dispersion, requires the knowledge of electrostatic potential distribution, which can then be used to calculate other quantities,

such as particle-particle interaction force. Features of particle interaction are of great importance for the stability and properties of colloidal dispersions. Accurate numerical modeling based on the Poisson-Boltzmann equation can provide important information on effective particle interaction in colloidal systems.

Bowen and Sharif [1997] presented a two-dimensional (axisymmetric) finite-element solution combined with adaptive mesh refinement. They used the Galerkin finite-element method with nine-node quadrilateral elements to discretize the Poisson-Boltzmann equation. The Newton-Raphson iterative scheme was used to handle the nonlinear hyperbolic sine term, while the Debye-Huckel solution was used as the initial guess Later, Bowen and Sharif [1998] applied the axisymmetric Poisson-Boltzmann equation solver to simulate the interactive force between two like-charge particles within a cylindrical pore with the same sign of charge on the surface. The results showed that at some inter-particle distance the two particles exhibit attractive forces, which is similar to that observed experimentally in Larsen and Grier [1997], although the magnitude is about one order different. It was then argued that in Ref. [Bowen and Sharif, 1998] the difference might be due to the fact that the wall is planar in the experiments [Larsen and Grier, 1997], whereas in the simulation it is cylindrical. For realistic applications like this, a three-dimensional simulation is required for accurate results. However, not only are 3D simulations extremely

time-consuming in practice, but developing efficient tools for them is also quite difficult.

Dyshlovenko [2001] also presented an axisymmetric Poisson-Boltzmann equation solver using finite-element method combined with adaptive mesh refinement. Instead of using nine-node quadrilateral elements, he used six-node triangular elements with a second-order shape function. Later, Dyshlovenko [2002] simulated the same problem as Bowen and Sharif [1998], but found no like-charge attractive force at any inter-particle distance in his simulation. In addition, Das and Bhattacharjee [2005] examined the effects of the roughness of the wall on the distribution of the electrostatic force using the two-dimensional finite-element method.

As shown in the above reviews, most existing studies use finite-element method with adaptive mesh refinement to solve the two-dimensional Poisson-Boltzmann equation. Very few studies have focused on developing a parallelized three-dimensional Poisson-Boltzmann equation solver even though the development of one would be valuable in understanding real physical phenomena. Two possible obstacles that have hindered the development of three-dimensional simulation are the time-demanding computation of a meaningful 3D simulation and the very complicated implementation of a parallel adaptive mesh refinement for a 3D unstructured mesh. Thus, it is important that to develop a parallelized three-dimensional

Poisson-Boltzmann equation solver using the Galerkin finite-element method using the first- and the second-order shape function combined with a parallel adaptive mesh refinement (PAMR) module.

1-3-2 Mesh Refinement

1-3-2-1 Adaptive Mesh Refinement

In previous section, we have briefly described three kinds of mesh refinement scheme: local polynomial-degree-variation (*p*-refinement*)*, mesh movement (*r*-refinement) and mesh enrichment (*h*-refinement). As described earlier in previous section, the first [Zienkiewicz *et. al*., 1983; Peano, 1976] and second [Brackbill, 1993; Mackenzie and Robertson, 2002] methods have some disadvantages. Thus, comparatively few studies were conducted along these directions. It is generally regarded that mesh enrichment method (*h-refinement*) outperforms the first two methods [Rausch et. al., 1991]. One of the most important advantages is that the mesh enrichment technique is in general many times faster and robust than the re-meshing technique [Connell and Holms, 1994]. However, some disadvantages were mentioned in Ref.[Rausch *et. al*., 1991]. For example, appearance of the hanging nodes induces significant modification to the numerical method which couples the mesh refinement scheme. Fortunately, this can be overcome by some simple methods through the

elimination of hanging nodes, such as that proposed by Kallinderis and Vijayan [1993]. In the current study, we intend to employ similar technique to improve the accuracy of the Poisson-Boltzmann equation solver.

1-3-2-2 Parallel Adaptive Mesh Refinement (PAMR)

For treating large-scale simulation problems, not only we have to parallelize the numerical solver, but also we have to parallelize the mesh refinement scheme. There are several ways to classify PAMR algorithm, and some of them are reviewed in the following.

First, from the viewpoint of practical implementation on computers, PAMR can be generally categorized into two types: vectorized shared memory and distributed memory. In the shared memory type [Waltz, 2004], the hardware for parallel implementation is generally very expensive, and the coding is comparably easy, like using OpenMP [Chandra *et. al*., 2001], for instance. Memory access is direct and fast, which makes communication cost among processors minimal. Moreover, load balancing among processors is expected to be good, while memory contention may occur which possibly slows down the speed. In contrast, the hardware is more affordable for the distributed memory type than the shared memory type, while the coding is rather involved using MPI [Karypis *et. al.*, 2003], for instance. Global data access is gained through network communication, which often becomes the bottleneck for better parallel performance if a large number of processors is used. However, this becomes increasingly unimportant because of the rapid increase in the

speed of network communication, and the dramatic reduction of costs in the past decade. Deciding on what type of parallel machine to implement greatly impacts the fundamental algorithm design as well as the practical implementation of the PAMR.

Second, from the viewpoint of the adaptive mesh refinement (AMR) algorithm itself, there are generally two kinds of mesh refinement: *h-refinement* and mesh regeneration. Mesh regeneration AMR [Wang and Harver, 1994] regenerates mesh from a mesh generator based on the distribution of error estimator obtained from the previous coarse mesh. For a large-scale 3-D computation, there exist two problems. First, the process often takes too much time, and second, the interpolation of previous solutions onto the new mesh is rather inefficient, which may slow down the convergence for the simulation on the new mesh. However, the data structure is relatively simple as compared to the *h-refinement* AMR. The *h-refinement* AMR adds grid points onto cell edges based on the error estimator obtained from the previous coarse mesh. It is generally much faster than the mesh regeneration type, and the interpolation onto the new mesh from the previous solution is straightforward, which may speed up the convergence for the computation on the new mesh. Often, the resulting "edge-based" data structure is very complicated and memory demanding, especially for the 3-D PAMR on the memory-distributed type of machine. Nevertheless, this type of AMR also has another advantage in that its data structure may be readily modified for mesh de-refinement, which may become important for a time-dependent simulation. Additionally, the *h-refinement* AMR possesses high spatial locality, which makes possible the parallel implementation on a memory-distributed machine using domain decomposition.

Third, from the viewpoint of the unstructured mesh solver itself, two types of

mesh data are required: the finite element type of connectivity (or connectivity) and cell-based connectivity (or cell neighbor-identifying information). The equation solver often needs the former, while the latter is required by some particle method for particle tracking, such as DSMC and PIC for solving the Boltzmann equation with neutral and charged species, respectively. However, the past development of the AMR scheme only produced the finite element type of connectivity. Related cell neighbor-identifying information can of course be obtained by post-processing serially (not in parallel) the node-based connectivity, while it may become time-consuming if the resulting mesh size is large. Thus, an ideal PAMR scheme may be expected to directly generate both these two sets of data.

For adaptive mesh refinement of three-dimensional tetartohedral mesh for the Poisson-Boltzmann equation, Cortis and Friesner [1997a, 1997b] proposed an automatic Finite-Element mesh generation system. Authors presented the algorithms of the construction of a tetrahedral mesh using a predetermined spatial distribution of vertices adapted to the geometry of the dielectric continuum solvent model. Besides, Baker[2000, 2001b] and his co-worker [Holst, 2000] publish the adaptive multilevel finite element treatment of the nonlinear PB equation using highly scalable parallel adaptive meshing methods which all of the numerical procedures are implemented in MANIFOLD CODE (MC) [web site, http://www.scicomp.ucsd.edu/ ~mholst/ codes/

mc/], a small self-contained parallel adaptive multilevel finite element software package.

1-4 Objectives of the Thesis

Based on previous reviews, the specific objectives of the thesis are summarized as follows:

1. To develop a parallelized 3-D Poisson-Boltzmann equation solver (PPBES) using finite element method with first- and second-order shape function on an unstructured tetrahedral mesh;
2. To develop a parallel adaptive mesh refinement (PAMR) for unstructured tetrahedral mesh;
3. To couple the PPBES and PAMR and apply it to compute a realistic 3-D problem.

1-5 Organization of the Thesis

The organization of this thesis is described as follows. Chapter 2 describes the numerical method for the parallelized Poisson-Boltzmann equation solver. Chapter 3 describes the validation and parallel performance study of the PPBES. Chapter 4 describes the proposed parallel adaptive mesh refinement, coupling scheme with

PPBES and application to a realistic 3-D problem. Finally, Chapter 5 describes the concluding remarks and recommendation of the future work. In addition, Appendix A describes the derivation of the nodal quadrature used in the present finite element formulation. Appendix B describes finite element formulation for hybrid mesh, which includes hexahedral, pyramid and tetrahedral mesh. Appendix C describes the first-order shape function of tetrahedral mesh and Appendix D describes the second-order shape function of tetrahedral mesh.

Chapter 2 Parallelized Poisson-Boltzmann Equation Solver (PPBES) Using Finite-Element Method

In this chapter, derivation of the Poisson-Boltzmann equation is first demonstrated, followed by the discretization of the Poisson-Boltzmann equation using Galerkin finite element method, introduction to general conjugate gradient method for solving linear algebraic equation, inexact Newton iterative scheme and the parallel implementation of the finite-element discretization is deliberated in turn. Finally, formula of calculating electrostatic force used in the present thesis is shown

2-1 Theoretical Model and Analysis EDL

Consider a liquid phase containing positive and negative ions in contact with a planar negatively charged surface. An EDL field near a charged surface is established. According to the theory of electrostatics, the relationship between the electrical potential ψ and the net charge density per unit volume ρ_e at any point in the solution is described by the three-dimensional Poisson's equation as

$$\nabla^2\psi = \frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} + \frac{\partial^2\psi}{\partial z^2} = -\frac{\rho_e}{\varepsilon\varepsilon_0} \tag{2-1}$$

where ε is dielectric constant of the solution and ε_0 is permittivity of vacuum. Applying the Boltzmann relation, assuming equilibrium, to the ionic number concentration distribution near a charged surface in a symmetric electrolyte solution (1:1), we have

$$n^+ = n_0 \exp(-\frac{z_i e\psi}{k_b T}) \quad \text{and} \quad n^- = n_0 \exp(\frac{z_i e\psi}{k_b T}) \tag{2-2}$$

Assuming the equilibrium Boltzmann distribution is reasonable, the number concentration of the type-*i* ion in a symmetric electrolyte solution is of the form

$$n_i = n_{i0} \exp(-\frac{z_i e\psi}{k_b T}) \tag{2-3}$$

where n_{i0} and z_i are bulk ionic concentration and the valence of type-*i* ions, respectively, e is the charge of a proton, k_b is the Boltzmann constant, T is the absolute temperature, and $\exp(\frac{z_i e\psi}{k_b T})$ is the Boltzmann factor. The Boltzmann distribution is applicable only when the system is in the thermodynamic equilibrium state. Then the net charge density in a unit volume of the fluid is given by:

$$\rho_e = ze(n^+ - n^-) = -2zen_0 \sinh(\frac{ze\psi}{k_b T}) \tag{2-4}$$

For a symmetric 1:1 electrolyte system, substituting Equation (2-4) in Equation (2-1), a nonlinear second-order three-dimensional Poisson-Boltzmann equation is obtained as

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} + \frac{\partial^2 \psi}{\partial z^2} = \frac{2zen_0}{\varepsilon\varepsilon_0} \sinh(\frac{ze\psi}{k_b T}) \tag{2-5}$$

By defining the Debye-Huckel parameter $\kappa^2 = \frac{2z^2 e^2 n_0}{\varepsilon\varepsilon_0 k_b T}$ and applying the dimensionless groups as follows,

$$\bar{x} = \kappa x, \quad \bar{y} = \kappa y, \quad \bar{z} = \kappa z \quad \text{and} \quad \bar{\psi} = \frac{ze\psi}{k_b T}$$

The Poisson-Boltzmann equation, equation (2-5), can be re-written as:

$$\frac{\partial^2 \bar{\psi}}{\partial \bar{x}^2} + \frac{\partial^2 \bar{\psi}}{\partial \bar{y}^2} + \frac{\partial^2 \bar{\psi}}{\partial \bar{z}^2} = \sinh(\bar{\psi}) \tag{2-6}$$

where $1/\kappa$ (Debye length λ_d) is the characteristic length of the EDL.

Generally, solving this equation with appropriate boundary conditions, the electrical potential distribution ψ of the EDL can be obtained, and the local charge density distribution ρ_e can then be determined from equation (2-4). It should be noted that the Debye-Huckel parameter κ^2 is independent of the solid surface properties and is determined by the liquid properties only, such as the electrolyte's valence and the bulk ionic concentration.

2-2 Discretization Using Finite Element Method with Tetrahedral Mesh

The FEM is the computer-aid mathematical technique for obtaining approximate numerical solution to the abstract of calculus that predicts the response of physical system subjected to the external influences.

Such problems arise in many areas of engineering, science, and applied mathematics. Applications to date have occurred principally in the areas of solid mechanics, heat transfer, fluid mechanics, and electromagnetism.

There are many salient features in FEM:

1. The domain is divided into smaller regions called elements. Adjacent elements touch without overlapping, and there are no gaps between the

elements. The shapes of the elements are intentionally made as simple as possible.

2. In each element the governing equations, usually in differential or variation (integral) form, are transformed into algebraic equation. The element equations are algebraically identical for all elements of the same type, which usually need to be derived for only one or two typical elements.

3. The resulting numbers are assembled (combined) into a much larger set of algebraic equations called the system equations. Such huge systems of equations can be solved economically because the matrix of coefficients is "spares".

4. Solved the matrix problem.

FEM seeks an approximate solution $\tilde{U}$, an explicit expression for U, in terms of known functions, which approximately satisfies the governing equations and boundary conditions. It obtains an approximate solution by using the classical trial-solution procedure. Normally, it is not possible to obtain strong solutions for the problem at hand. Instead one usually decretive the otherwise continuous problem and obtain so-called weak solutions; weak in the sense of approximation.

The application of the finite element method to a given problem involves

the following six steps [Thompson, 2003; Zienkiweicz, 2000]:

1. Development of element equation.
2. Discretization of solution domain into a finite element mesh.
3. Assembly of element equations.
4. Introduction of boundary conditions.
5. Solution for nodal unknowns.
6. Computation of solution and related quantities over each element.

2-2-1 Interpolation with First-order Shape Function

In this section, we will describe the FE discretization of the three-dimensional nonlinear Poisson-Boltzmann equation using first-order shape function in detail. In general, the procedure are divided into the following steps.

Step1:

We use $\tilde{\psi}$ to approximate $\bar{\psi}$, then we set the trial solution:

$$\tilde{\psi}^{(e)} = N_1^{(e)} a_1 + N_2^{(e)} a_2 + N_3^{(e)} a_3 + N_4^{(e)} a_4 = \sum_{j=1}^{4} N_j^{(e)}(x,y,z) a_j \qquad (2\text{-}7)$$

where x, y, z are the independent variables in the problems. The functions $N_j(x,y,z)$ are known functions called trial functions (basis).

The purpose is to determine specific numerical values for each of the parameters a_j (such as potential). Then, using the Galerkin method, for each parameter a_j we

require that a weighted average of $R(x,y,z)$ over the entire domain to be zero. The weighting functions are trial functions $N(x,y,z)$ associated with each a_j.

Step2:

Applying Galerkin residual method, from Equation (2-6):

$$\oint_\Omega N_i^{(e)}\left(\frac{\partial^2\overline{\psi}}{\partial\overline{x}^2}+\frac{\partial^2\overline{\psi}}{\partial\overline{y}^2}+\frac{\partial^2\overline{\psi}}{\partial\overline{z}^2}\right)d\Omega=\oint_\Omega N_i^{(e)}\sinh(\overline{\psi})d\Omega \tag{2-8}$$

the Galerkin residual method is a weighting residual method, which uses the trial function as the weighting function. Thus, $N_i^{(e)}$ is not only a weighting function, but also a shape function. Note the integral $\oint_\Omega d\Omega$ means integrate over the volume of element. In the present study, we employ tetrahedral elements, which have four degree of freedom (DOF) in each element.

Within the tetrahedral element, as shown in Fig.2-1, the function $\widetilde{\psi}_i^e$, the first-order shape function, can be approximated as,

$$\widetilde{\psi}_i^{(e)}(x,y,z)=a_i^{*(e)}+b_i^{*(e)}x+c_i^{*(e)}y+d_i^{*(e)}z \quad , \ i=1\sim 4 \tag{2-9}$$

The coefficients $a_i^{*(e)}$, $b_i^{*(e)}$, $c_i^{*(e)}$ and $d_i^{*(e)}$ can be determined by enforcing Equation (2-9) at the four nodes of the element a unit value at each of the vertex. From which we can obtain,

$$\begin{aligned} a^{*(e)} &= \frac{1}{6V^{(e)}}\begin{vmatrix} \phi_1^{(e)} & \phi_2^{(e)} & \phi_3^{(e)} & \phi_4^{(e)} \\ x_1^{(e)} & x_2^{(e)} & x_3^{(e)} & x_4^{(e)} \\ y_1^{(e)} & y_2^{(e)} & y_3^{(e)} & y_4^{(e)} \\ z_1^{(e)} & z_2^{(e)} & z_3^{(e)} & z_4^{(e)} \end{vmatrix} \\ &= \frac{1}{6V^{(e)}}(a_1^{*(e)}\phi_1^{(e)}+a_2^{*(e)}\phi_2^{(e)}+a_3^{*(e)}\phi_3^{(e)}+a_4^{*(e)}\phi_4^{(e)}) \end{aligned} \tag{2-10}$$

$$b^{*(e)} = \frac{1}{6V^{(e)}}\begin{vmatrix} 1 & 1 & 1 & 1 \\ \phi_1^{(e)} & \phi_2^{(e)} & \phi_3^{(e)} & \phi_4^{(e)} \\ y_1^{(e)} & y_2^{(e)} & y_3^{(e)} & y_4^{(e)} \\ z_1^{(e)} & z_2^{(e)} & z_3^{(e)} & z_4^{(e)} \end{vmatrix} \tag{2-11}$$

$$= \frac{1}{6V^{(e)}}(b_1^{*(e)}\phi_1^{(e)} + b_2^{*(e)}\phi_2^{(e)} + b_3^{*(e)}\phi_3^{(e)} + b_4^{*(e)}\phi_4^{(e)})$$

$$c^{*(e)} = \frac{1}{6V^{(e)}}\begin{vmatrix} 1 & 1 & 1 & 1 \\ x_1^{(e)} & x_2^{(e)} & x_3^{(e)} & x_4^{(e)} \\ \phi_1^{(e)} & \phi_2^{(e)} & \phi_3^{(e)} & \phi_4^{(e)} \\ z_1^{(e)} & z_2^{(e)} & z_3^{(e)} & z_4^{(e)} \end{vmatrix} \tag{2-12}$$

$$= \frac{1}{6V^{(e)}}(c_1^{*(e)}\phi_1^{(e)} + c_2^{*(e)}\phi_2^{(e)} + c_3^{*(e)}\phi_3^{(e)} + c_4^{*(e)}\phi_4^{(e)})$$

$$d^{*(e)} = \frac{1}{6V^{(e)}}\begin{vmatrix} 1 & 1 & 1 & 1 \\ x_1^{(e)} & x_2^{(e)} & x_3^{(e)} & x_4^{(e)} \\ y_1^{(e)} & y_2^{(e)} & y_3^{(e)} & y_4^{(e)} \\ \phi_1^{(e)} & \phi_2^{(e)} & \phi_3^{(e)} & \phi_4^{(e)} \end{vmatrix} \tag{2-13}$$

$$= \frac{1}{6V^{(e)}}(d_1^{*(e)}\phi_1^{(e)} + d_2^{*(e)}\phi_2^{(e)} + d_3^{*(e)}\phi_3^{(e)} + d_4^{*(e)}\phi_4^{(e)})$$

where

$$V^{(e)} = \frac{1}{6}\begin{vmatrix} 1 & 1 & 1 & 1 \\ x_1^{(e)} & x_2^{(e)} & x_3^{(e)} & x_4^{(e)} \\ y_1^{(e)} & y_2^{(e)} & y_3^{(e)} & y_4^{(e)} \\ z_1^{(e)} & z_2^{(e)} & z_3^{(e)} & z_4^{(e)} \end{vmatrix} = \text{volume of the element} \tag{2-14}$$

Thus, the coefficients $a_i^{*(e)}$, $b_i^{*(e)}$, $c_i^{*(e)}$ and $d_i^{*(e)}$ can be easily determined from expansion of the determinants and are not listed here for brevity. They can be found in Appendix C.

Substituting the expression for $a^{*(e)}$, $b^{*(e)}$, $c^{*(e)}$ and $d^{*(e)}$ back into Equation (2-9), we obtain the interpolation function as given by:

$$\mathrm{N}_{\mathrm{j}}^{(\mathrm{e})}(x, y, z) = \frac{1}{6V^{(e)}}(a_j^{*(e)} + b_j^{*(e)}x + c_j^{*(e)}y + d_j^{*(e)}z) \tag{2-15}$$

It can be easily shown that the derived shape function has the following property

as it should be.

$$\mathrm{N}_i^{(e)}(x,y,z)=\delta_{ij}=\begin{cases}1 & i=j\\ 0 & i\neq j\end{cases} \tag{2-16}$$

and furthermore that $\mathrm{N}_i^{(e)}(x,y,z)$ vanishes when the observation point is on the surface of the tetrahedron opposite the j-th node. As a result, inter-element continuity of the interpolated function is guaranteed.

Step3:

Integrating Equation (2-8) by part, we can obtain

$$\begin{aligned}&\sum_i\iiint[\frac{\partial}{\partial x}(\frac{\partial\overline{\psi}^{(e)}}{\partial x}N_i^{(e)})+\frac{\partial}{\partial y}(\frac{\partial\overline{\psi}^{(e)}}{\partial y}N_i^{(e)})+\frac{\partial}{\partial z}(\frac{\partial\overline{\psi}^{(e)}}{\partial z}N_i^{(e)})]dxdydz\\&-\sum_i\iiint[(\frac{\partial\overline{\psi}^{(e)}}{\partial x}\frac{\partial N_i^{(e)}}{\partial x})+(\frac{\partial\overline{\psi}^{(e)}}{\partial y}\frac{\partial N_i^{(e)}}{\partial y})+(\frac{\partial\overline{\psi}^{(e)}}{\partial z}\frac{\partial N_i^{(e)}}{\partial z})]dxdydz\\&=\sum_i\iiint N_i^{(e)}\sinh(\overline{\psi}^{(e)})dxdydz\end{aligned} \tag{2-17}$$

Step4:

Applying Gauss divergence theorem to Equation (2-17), we can obtain

$$\begin{aligned}&\sum_j\oiint[\frac{\partial\overline{\psi}}{\partial x}\hat{i}+\frac{\partial\overline{\psi}}{\partial y}\hat{j}+\frac{\partial\overline{\psi}}{\partial z}\hat{k}]N_i^{(e)}\cdot\hat{n}dxdy\\&-\sum_i\iiint[(\frac{\partial\overline{\psi}^{(e)}}{\partial x}\frac{\partial N_i^{(e)}}{\partial x})+(\frac{\partial\overline{\psi}^{(e)}}{\partial y}\frac{\partial N_i^{(e)}}{\partial y})+(\frac{\partial\overline{\psi}^{(e)}}{\partial z}\frac{\partial N_i^{(e)}}{\partial z})]dxdydz\\&=\sum_i\iiint N_i^{(e)}\sinh(\overline{\psi}^{(e)})dxdydz\end{aligned} \tag{2-18}$$

Then, by substituting trial function, Equation (2-7), into Equation (2-18), we can obtain

$$\sum_j \sum_i \oiint [\frac{\partial N_j^{(e)} a_j}{\partial x}\hat{i} + \frac{\partial N_j^{(e)} a_j}{\partial y}\hat{j} + \frac{\partial N_j^{(e)} a_j}{\partial z}\hat{k}] N_i^{(e)} \cdot \hat{n} dxdy$$
$$- \sum_j \sum_i \iiint [(\frac{\partial N_j^{(e)} a_j}{\partial x}\frac{\partial N_i^{(e)}}{\partial x}) + (\frac{\partial N_j^{(e)} a_j}{\partial y}\frac{\partial N_i^{(e)}}{\partial y}) + (\frac{\partial N_j^{(e)} a_j}{\partial z}\frac{\partial N_i^{(e)}}{\partial z})] dxdydz \quad (2\text{-}19)$$
$$= \sum_i \iiint N_i^{(e)} \sinh(\tilde{\psi}^{(e)}) dxdydz$$

Now define:

$$\hat{\tau} = -(\frac{\partial N_j^{(e)} a_j}{\partial x}\hat{i} + \frac{\partial N_j^{(e)} a_j}{\partial y}\hat{j} + \frac{\partial N_j^{(e)} a_j}{\partial z}\hat{k}) \quad (2\text{-}20)$$

Equation (2-19) can be rewritten as follows,

$$\sum_j \sum_i \iiint [(\frac{\partial N_j^{(e)}}{\partial x}\frac{\partial N_i^{(e)}}{\partial x}) + (\frac{\partial N_j^{(e)}}{\partial y}\frac{\partial N_i^{(e)}}{\partial y}) + (\frac{\partial N_j^{(e)}}{\partial z}\frac{\partial N_i^{(e)}}{\partial z})] a_j dxdydz$$
$$= -\sum_i \iiint N_i^{(e)} \sinh(\tilde{\psi}^{(e)}) dxdydz - \oiint \tau_n dxdy \quad (2\text{-}21)$$

or,

$$\sum_{j=1}^{4} \sum_{i=1}^{4} \iiint \frac{1}{36V^{(e)^2}} (b_j^{*(e)} b_i^{*(e)} + c_j^{*(e)} c_i^{*(e)} + d_j^{*(e)} d_i^{*(e)}) a_j dxdydz$$
$$= -\sum_{i=1}^{4} \iiint N_i^{(e)} \sinh(\tilde{\psi}^{(e)}) dxdydz - \oiint \tau_n dxdy \quad (2\text{-}22)$$

where

$$\iiint dxdydz = \mathrm{V}^{(e)} \quad (2\text{-}23)$$

Next, by substituting Equation (2-23) into Equation (2-22), we can obtain

$$\sum_{j=1}^{4} \sum_{i=1}^{4} \frac{1}{36V^{(e)}} (b_j^{*(e)} b_i^{*(e)} + c_j^{*(e)} c_i^{*(e)} + d_j^{*(e)} d_i^{*(e)}) a_j$$
$$= -\sum_{i=1}^{4} \iiint N_i^{(e)} \sinh(\tilde{\psi}^{(e)}) dxdydz - \oiint \tau_n dxdy \quad (2\text{-}24)$$

Define

$$K_{ij}^{(e)} = \sum_{j=1}^{4} \sum_{i=1}^{4} \frac{1}{36V^{(e)}} (b_j^{*(e)} b_i^{*(e)} + c_j^{*(e)} c_i^{*(e)} + d_j^{*(e)} d_i^{*(e)}) \quad (2\text{-}25)$$

$$F_i^{(e)} = -\sum_{i=1}^{4} \iiint N_i^{(e)} \sinh(\tilde{\psi}^{(e)}) dxdydz - \oiint \tau_n dxdy \quad (2\text{-}26)$$

Finally, we can obtain the matrix equation of the FE-discretized Poisson-Boltzmann Equation as

$$[K_{ij}^{(e)}]\{a_j^{(e)}\} = \{F_i^{(e)}\} \tag{2-27}$$

2-2-2 Interpolation with Second-order Shape Function

For some physical problems, tetrahedral elements with first-order shaper function are good enough for maintaining high accuracy. However, it may not be so for some problems like the present Poisson-Boltzmann equation due to the exponentially varying source term. Elements with higher order shape function are often necessary to improve the accuracy of the numerical solution, while the complexity of implementation and required memory increases with increasing order of shape function. Considering the compromise between numerical accuray and complexity of implementation, elements with the second-order shape function represents an optimum choice. Although the 4-node tetrahedron is the simplest solid element, they are not the preferred elements in engineering analysis because of their low accuracy. In what follows, elements with the second-order shape function are presented in detail.

The natural or tetrahedral coordinates system L_1, L_2, L_3 and L_4 of a tetrahedron element were shown in Fig. 2-1. For a quadratic interpolation model,

there are 10 nodal unknowns, 1 at each of the nodes as indicated in Fig. 2-2. The nodes 1, 2, 3 and 4 correspond to the corners, whereas the nodes 5 to 10 are located at the midpoints of the edges of the tetrahedron element. The variation of the field variable is given by

$$\phi(x,y,z) = [N]\vec{\Phi}^{(e)} = [N_1 \quad N_2 \quad \ldots \quad N_{10}]\vec{\Phi}^{(e)} \tag{2-28}$$

where N_i can be found as

$$\begin{cases} \mathrm{N}_i = L_i(2L_i - 1), & i = 1, 2, 3, 4 \\ N_5 = 4L_1L_2 \\ N_6 = 4L_2L_3 \\ N_7 = 4L_1L_3 \\ N_8 = 4L_2L_4 \\ N_9 = 4L_3L_4 \\ N_{10} = 4L_1L_4 \end{cases} \tag{2-29}$$

and

$$\begin{aligned} \vec{\Phi}^{(e)} &= \begin{Bmatrix} \Phi_1 \\ \Phi_2 \\ \vdots \\ \Phi_{10} \end{Bmatrix}^{(e)} = \begin{Bmatrix} \phi(x_1, y_1, z_1) \\ \phi(x_2, y_2, z_2) \\ \vdots \\ \phi(x_{10}, y_{10}, z_{10}) \end{Bmatrix}^{(e)} \\ &= \begin{Bmatrix} \phi(at\ L_1 = 1, L_2 = L_3 = L_4 = 0) \\ \phi(at\ L_2 = 1, L_1 = L_3 = L_4 = 0) \\ \vdots \\ \phi(at\ L_3 = L_4 = \frac{1}{2}, L_1 = L_2 = 0) \end{Bmatrix} \end{aligned} \tag{2-30}$$

Note that L_j is defined the same as Equation (2-15), but N_i are replaced with L_j. For example, to assemble the coefficient matrix $\iiint \nabla\phi_1 \cdot \nabla\phi_1 dxdydz$,

$$\nabla\phi_1 = \frac{1}{18V^2}\begin{Bmatrix} [(2b_1^{*2}x) - 3V\cdot b_1^* + 2a_1^*b_1^* + 2b_1^*c_1^*y + 2b_1^*d_1^*z]\vec{i} \\ +[(2c_1^{*2}y) - 3V\cdot c_1^* + 2a_1^*c_1^* + 2b_1^*c_1^*y + 2c_1^*d_1^*z]\vec{j} \\ +[(2d_1^{*2}z) - 3V\cdot d_1^* + 2a_1^*d_1^* + 2b_1^*d_1^*y + 2c_1^*d_1^*y]\vec{k} \end{Bmatrix}$$

$$= \frac{1}{3V}\begin{Bmatrix} b_1^*[-\frac{1}{2} + 2L_1]\vec{i} \\ +c_1^*[-\frac{1}{2} + 2L_1]\vec{j} \\ +d_1^*[-\frac{1}{2} + 2L_1]\vec{k} \end{Bmatrix} \tag{2-31}$$

and

$$\nabla\phi_1\cdot\nabla\phi_1 = \frac{(b_1^{*2} + c_1^{*2} + d_1^{*2})(-\frac{1}{2} + 2L_1)^2}{9v^2}$$
$$= \frac{(b_1^{*2} + c_1^{*2} + d_1^{*2})}{9v^2}(4L_1^2 - 2L_1 + \frac{1}{4}) \tag{2-32}$$

By employing area coordinates, the volume integrals can be easily evaluated from the relation

$$\int_V L_1^a L_2^b L_3^c L_4^d dv = \frac{a!\,b!\;\,c!\,d!}{(3+a+b+c+d)!}6V \tag{2-33}$$

Then,

$$\iiint \nabla\phi_1\cdot\nabla\phi_1 dxdydz = \frac{(b_1^{*2} + c_1^{*2} + d_1^{*2})}{9v}\frac{3}{20} \tag{2-34}$$

Some discussion of the derivation are shown in Appendix D and are not described here for brevity.

2-3 Conjugate Gradient Method for Linear Algebra Equation

The conjugate gradient method was developed in 1952 by Hestenes and Stiefel [1952] as an improvement to the steepest descent method. Whereas steepest descent

approaches the solution asymptotically, the conjugate gradient method will find the solution in n iterations (assuming no round-off error). There are many solvers including this method or subroutine presently [Holst et. al., 2000]. In our laboratory, the subroutine of conjugate gradient method of parallel version has developed by Hsu, K.-H [2006].

Generally, in the linear system $Ax = b$ suppose the coefficient matrix A is symmetric and positive definite. Solving $Ax = b$ is equivalent to minimizing the quadratic function $Q(x) = \frac{1}{2} x^T Ax - x^T b$. The conjugate gradient method is based on the idea that the convergence to the solution could be accelerates if we minimize Q over just the line that points down gradient. To determine x^{i+1} we minimize Q over $x^0 + span(p^0, p^1, p^2, \cdots, p^i)$, where the p^k represent previous search directions. An added advantage to this approach is that, if we can select the p^k to be linearly independent, then the dimension of the hyperplane will grow one dimension with each iteration of the conjugate gradient method. This would imply that (assuming infinite precision arithmetic) the solution of the linear system $Ax = b$ would be obtained in no more than N steps, where N is the number of unknowns in the system. Major algorithm in this method is summarized as:

For each process j;

step 1. $b_{I\,j} = Update(b_{I\,j})$

step 2. $x_j^0 = 0,\ \ x_{Ij}^0 = 0;\ \ r_j^0 = b_j$ and $r_{Ij}^0 = b_{Ij}$

step 3. Solve $M(z_j^0, z_{Ij}^0) = (r_j^0, r_{Ij}^0);\ \ p_j^0 = z_j^0,\ \ p_{Ij}^0 = z_{Ij}^0$

step 4. $r^0 = Inner\ \mathrm{Pr}oduct(z_j^0, z_{Ij}^0; r_j^0, r_{Ij}^0)$

Do k = 0, 1, 2…., Maximum Iterations

step 5. $(q_j^k, q_{Ij}^k) = Matrix\ Multiply(A, (p_j^k, p_{Ij}^k))$

step 6. $\tau^k = Inner\ \mathrm{Pr}oduct(p_j^0, p_{Ij}^0; q_j^0, q_{Ij}^0)$

step 7. $\alpha^k = r^k \big/ \tau^k$

step 8. $x_j^{k+1} = r_j^k + \alpha^k p_j^k;\ \ x_{Ij}^{k+1} = r_{Ij}^k + \alpha^k p_{Ij}^k$

step 9. $r_j^{k+1} = r_j^k - \alpha^k q_j^k;\ \ r_{Ij}^{k+1} = r_{Ij}^k - \alpha^k q_{Ij}^k$

step 10. Solve $M(z_j^{k+1}, z_{Ij}^{k+1}) = (r_j^{k+1}, r_{Ij}^{k+1})$

step 11. $r^{k+1} = Inner\ \mathrm{Pr}oduct(z_j^{k+1}, z_{Ij}^{k+1}; r_j^{k+1}, r_{Ij}^{k+!})$

step 12. If ($\sqrt{r^{k+1}} \le Tolerance$) END

step 13. $\beta^k = r^{k+1} \big/ r^k;\ \ p_j^{k+1} = z_j^{k+1} + \beta^k p_j^k,\ \ p_{Ij}^{k+1} = z_{Ij}^{k+1} + \beta^k p_{Ij}^k$

END DO

For parallel conjugate gradient method, in step 1 and step 5, the node-node of interface of subdomain need to do a communication. Besides, in step 6, all node of subdomain need to do one collective communication.

2-4 Inexact Newton-Raphson Iterative Scheme

The method of Newton-Raphson is a classical tool for the study of the existence of the solution of nonlinear PDE of certain types [Holst *et. al*., 1994]. It is also useful for the numerical solution of nonlinear types of problems approximated, for instance, by the finite difference, finite element or boundary element method. In the present thesis, an inexact Newton iterative scheme is employed to solve the nonlinear FE-discretized Poisson-Boltzmann equation.

Following the finite element formulation, Equation (2-6) yields a nonlinear algebraic system in the form

$$[K]\{U\} = \{F(U)\}, \tag{2-35}$$

where $[K]$ and $\{F(U)\}$ are derived from the Laplacian term and the nonlinear forcing term, respectively. Note that the additional term due to the non-homogenous Dirichlet boundary condition included in $\{F(U)\}$. This nonlinear algebraic system is solved by an inexact Newton iterative method, which can be described as follows. Let $U^{(0)}$ be a given initial estimate and assume that $U^{(k)}$ is the current approximation. Then the new approximation $U^{(k+1)}$ can be computed through

$$A^{(k)}U^{(k+1)} = B(U^{(k)}), \tag{2-36}$$

where $A^{(k)} = [K] - J^{(k)}$ and $B(U^{(k)}) = F(U^{(k)}) - J^{(k)}U^{(k)}$. Equation (2-36) is solved at each iterative step until convergence is reached. In our implementation, the non-zero

entries of $A^{(k)}$ are stored in the compressed sparse row (CSR) format that is computationally efficient both in terms of storage and matrix operations [Saad, 2003]. Here, $J^{(k)}$ is the Jacobian matrix for $F(U)$. Further, it can be readily shown that $J^{(k)}$ is a diagonal matrix if the nodal quadrature is used to evaluate the integration of the nonlinear term in the Galerkin FE formulation. Details of the derivation are shown in Appendix A and are not described here for brevity. Diagonalization of the Jacobian matrix $J^{(k)}$ is important since the memory for storage is greatly reduce as well as the much easier parallelization of this matrix in practice.

2-5 Parallel Implementation of the P-B Equation Solver

2-5-1 Introduction to Parallel Computing

There is a continual demand for greater computational power from computer system than is currently possible. Areas requiring great computational speed include numerical simulation of scientific and engineering problem. A parallel computer, as defined by Almasi and Gottlieb [1989], is a collection of processing element that cooperate and communicate to solve large problems fast. Parallel computers can be viewed as a collection of processors and memory units which are connected by an interconnection network. The purpose of parallel computing is to get work done in less time and also to solve problems which can not fit in a single computers memory.

A parallel computer, as we have mentioned is either a single computer with multiple internal processors or multiple computer interconnected to from a coherent high-performance computing platform. There are two basic types of parallel computer: Shared memory multiprocessor and Distributed-memory multicomputer.

A natural way to extend the single-processor model is to have multiple processors connected to multiple memory modules, such that each processor can access any memory module in a so-call shared memory configuration, as shown in Figure 2-3. The connection between the processors and memory is through some from of interconnection network. A share memory multiprocessor system employs a single address space, which means that each location in the whole main memory system has a unique address that is used by each processor to access the location.

In Distributed Memory architecture parallel computers, memory is physically distributed among processors; each local memory is directly accessible only by its processor. Synchronization is achieved by moving data between processors by a fast interconnection network shown in Fig 2-4.

In our laboratory, we had set up PC Clusters which are use distributed memory system shown in Fig 2-5. The advantage of this kind of architecture is that it is scalable to a large number of commodity processors. A major concern with this scheme is data decomposition; the data being operated should be divided equally to

balance the load, and also should be done in an efficient manner in order to minimize communication. The scalability depends on the type of interconnection between the processors.

2-5-2 Parallel Implementation

In order to parallelize the Poisson-Boltzmann equation solver, we chose Message Passing Interface (MPI) library [http://www-unix.mcs.anl.gov/mpi/] as our standard parallel programming. MPI provides library routines for message-passing and associated operations. A fundamental aspect of MPI is that it defines a standard but not the implementation, just as programming languages are defined but not how to compilers for the languages are implemented. MPI library of a set of standard subroutine calls which allow parallel programs to be written in distributed memory system.

Besides, we use METIS [http://glaros.dtc.umn.edu/gkhome/views/metis] to distribute the cells and nodes in a system. METIS is a family of programs for partitioning unstructured graphs and hypergraphs and computing fill-reducing orderings of sparse matrices. The underlying algorithms used by METIS are based on the state-of-the-art multilevel paradigm that has been shown to produce high quality results and scale to very large problems.

The Poisson-Boltzmann equation solver is parallelized by domain decomposition as shows Fig.2-6. In distributed parallel finite element analysis, the domain considered is first decomposed into a number of subdomains by preprocessing program. Then, one processor is assigned for each subdomain, though other variations are also possible. Computations are then performed on each processor on the local subdomain and communication takes place with the other processors whenever needed, for example, during the matrix solving process. If the domain is partitioned into a set of non-overlapping subdomains then the methods used are called iterative substructuring methods.

Fig 2-7 summarizes schematically the procedure of the proposed parallelized Poisson-Boltzmann equation solver. The steps of the procedure are briefly described here.

1. Initialize MPI and synchronize all processors to prepare for parallel computation.
2. Read in the grid from the host processor and distribute the grid to all processors according to pre-partitioned information.
3. Compute the coefficient matrix in each processor and make any necessary communication between neighboring processors.
4. Set up initial and boundary conditions in each processor and set up k=0.

5. Newton iteration loop begins, k=k+1.

6. Update [A], {F} in equation (2-35) and make any necessary communication between neighboring processors.

7. Solve $A^{(k)}U^{(k+1)} = F(U^{(k)})$ using Parallel CG with the subdomain-by-subdomain scheme.

8. If $\|\varepsilon\|_\infty \le \varepsilon_0$, then go to the next step. Otherwise, go back to **Step 5**. Note ε_0 is the prescribed convergence criterion of the Newton-Raphson scheme, while $\|\varepsilon\|_\infty = \max\{\left|\frac{U^{k+1} - U^k}{U^{k+1}}\right|\}$.

9. Gather all computed data in the host and calculate the distribution of the electric field.

10. Output the data and stop the program.

2-6 Force Calculation in an Electrostatic Field

From the potential distribution obtained by solving the Poisson-Boltzmann equation, the electrostatic force on the spherical particles is calculated by integrating the total stress tensor, defined as

$$T_{ij} = \Delta\Pi + \varepsilon[EE - \frac{1}{2}E \cdot EI] \tag{2-37}$$

where $E = -\nabla\psi$ is the electric field vector; I represents the identity tensor, and $\Delta\Pi$ is the osmotic pressure difference between the electrolyte at the particle surface

and bulk solution, $\Delta\Pi$ can defined as

$$\Delta\Pi = 2n_b KT(\cosh\psi - 1) \tag{2-38}$$

where n_b is the bulk ion number density of the electricity neutral electrolyte.

Force is obtained by integrating the stress tensor over the surface of a particle as

$$F = \int_S T_{ij} \cdot n dS \tag{2-39}$$

where n is the unit outward surface normal, and the subscript S represents the area over the closed surface of the particle. Note that the stress tensor is calculated from the potential but the force on the sphere also includes a contribution from the osmotic pressure generated by the electric field. However, pressure variation due to the field at the sphere surface is normally canceled by electrical effects [Russel *et. al*., 1989] so that we can assume the $\Delta\Pi$ term is zero.

Chapter 3 Validation of the Parallel Poisson-Boltzmann Equation Solver

3-1 Convergence of the PPBES

Table 3-1 (a) and (b) summarizes the convergence study with a good initial guess and all-zero guess, respectively for a typical simulation (CPU No.=6; 81,354 nodes and 458,064 elements). A good initial guess is from the solution of last level mesh by interpolation. Results show that, with a good initial guess, only three Newton iterations are required to achieve the convergence criterion (ε_0=10^{-7}) in the present inexact Newton iterative scheme. On the other hand, if all zeros are used as the initial estimate, five Newton iterations are needed. Compared with all zeros are used as the initial estimate (relative residual=10^{-7}, total time=394.32sec), a good initial guess (relative residual=10^{-7}, total time=301.57sec) is saving about 24% total CPU time.

In addition, approximately 900-1300 iterations are required for the convergence (relative residual of from 10^{-5} to 10^{-7}) in the parallel CG solver within each Newton's iteration, since we do not use any preconditioning techniques. In addition, numerical experiments show that relative residual of 10^{-5} for the CG solver is the optimum choice in terms of the runtime and accuracy. If we choose a low relative residual (10^{-4}), not only increase the number of iterations for the CG solver but also bring the solution of inaccuracy. To further reduce runtime, a robust preconditioner, such as the

geometrical additive Schwarz domain decomposition method [Nadeem and Jimack, 2002], could be used. This work is currently in progress in our group and will be reported in the near future.

3-2 Validation 1: The Potential Distribution around a Charged Sphere

In presently papers review, Tuinier [2003] provided simple expressions are presented that give a very accurate description of electrostatic potential around a single sphere and cylinder for arbitrary double-layer thickness and surface potential. Completed PPBES is used to compute the potential distribution around a charged sphere, and the results are compared with analytical and approximate solutions [Tuinier, 2003] to validate the parallelized code.

Table 3-2 summarizes the previously developed analytical and approximate solutions for this case. Normalized radius of the sphere is kept as 5.0, while two cases of constant value of electrostatic potential (Ψ_0=1.0 and 5.0) are set at the surface of the charged sphere. Note that since PPBES is a 3D code, only 1/16 of the full domain is used for this simulation by taking advantage of the symmetry of the problem, while the Neumann boundary conditions are used at the symmetric planes. Fig 3-1 shows the surface mesh distribution of the sphere (176,309 elements; 36,396 nodes).

Fig. 3-2a shows that when Ψ_0=1.0, the computed potential distribution agrees very

well with both the analytical and approximate solutions [Tuinier, 2003], because at this lower Ψ_0, the Poisson-Boltzmann equation can be linearized correctly. However, Fig. 3-2b shows that when Ψ_0=5.0, the computational potential distribution deviates considerably from the analytical solution but stays quite close to the approximate solution. This is because at this higher Ψ_0, the Poisson-Boltzmann equation cannot be linearized correctly in the derivation of the analytical solution.

3-3 Validation 2: Interaction Between Two Like-charged Spheres within a Cylindrical Pore

In this section, we will simulate a geometrically confined pair of charged spheres concerning the phenomenon of long-range electrostatic attraction between particles of like charge [Larsen and Grier, 1997]. The long-range electrostatic interaction of two colloidal particles confined in a cylindrical pore was studied in the present paper [Bowen and Sharif, 1998], we use the same values of parameters and geometry, such as the 1:1 electrolyte, the potential on surfaces of the sphere particles $\Psi_s = 3.0$, the potential on the cylindrical pore surface $\Psi_p = 5.0$, the radius of the particles $a = 1.185$, the sphere radius to pore radius ratio 0.13. Taking advantage of the symmetry in this problem, only 1/4 of the whole physical domain is necessary for simulation, resulting in the need for 0.64 million elements (close to 0.11 million nodes).

Fig 3-3 shows the calculated potential distribution for the sphere confined in a pore at separation distances $r = 6$. Fig 3-4 shows the potential along the midplane BC at $r = 6$ for the first-order shape function. We can see the potential along this plane is almost decreasing at all distance, but after Z=3.4 the potential become increasing. By comparing with previous numerical data [Brwen and Sharif, 1998], we also have almost the same results, but still have 1% inaccuracy. Our calculations confirm speculation that the attraction between confined spheres. Besides, if we choose the second-order shape function, the accuracy will be improved.

3-4 Parallel performance of the PPBES

To study the parallel performance of the PPBES, we consider a test case with a charged sphere (radius=5, Ψ_0=2.0) confined in a cylindrical pore with the sphere radius to pore radius ratio of λ=0.35. Taking advantage of the symmetry in this problem, only 1/16 of the whole physical domain is necessary for simulation, resulting in the need for 0.61 million elements (close to 0.1 million nodes). Fig. 3-5 illustrates the resulting speedup as a function of the number of parallel processors (up to 32), and Table 3-3 summarizes the detailed time breakdown for various components of the parallelized code. Among these, the matrix solver using parallel CG method occupies ~50% of the total runtime and setup the matrix occupies ~8% for the number of processors tested in

the present study. This indicates that further reduction of the runtime can be highly expected if the matrix solver is greatly improved. Total runtime is ~2,486 seconds with the use of a single processor as compared to ~102 seconds with the use of 32 processors, which results in 76.2% in parallel efficiency. The results clearly show that the current parallel implementation of the Poisson-Boltzmann equation using the subdomain-by-subdomain method performs reasonably well for the typical problem size. A smaller problem size was not tested in this study since it is irrelevant in practice for typical three-dimensional applications. Indeed, it is expected that the parallel speedup will be even greater if a larger problem size is considered. Thus, not only can the current parallel implementation help to greatly reduce the runtime required for parametric study to understand the distribution of electrostatic potential but also increase tremendously the size of the problem that the memory-distributed parallel machine can handle.

3-5 The Second-order Shape Function for PPBES

Figure 3-6 shows that the distribution of potential around the charged spheres (radius a=0.325μm) near a like-charged plate (distance h = 2.5μm) at level-0 for solutions of the first-order shape function (Fig 3-6a) and the second-order shape function (Fig 3-6b). For the test cases, Table 3-4 shows that the mesh levels of the

number of nodes and elements. We can find the solution of second-order shape function better than the solution of first-order shape function at level-0 show in Figure 3-7 and Figure 3-8. It is a clear demonstration that when we use the second-order shape function for mesh of level-0, which the solution between the first-order shape function for mesh of level-3 and mesh of level-4. In other words, the solution of the second-order shape function will increase in accuracy over the first-order shape function. The advantage of second-order shape function is that they can define the edge points of elements in order to increase the solution of accuracy. It is not necessary to use very small elements. Therefore, we can use the second-order shape function to reduce the number of elements at refinement level.

Table 3-5 shows the information (case from Section 3-3) that compares the first-order with the second-order shape function for mesh of level 0. For the same order nodes (~110,000 nodes, the second-order shape function is from original 16,280 nodes), we can find the total CPU time is almost closed (~90 second). Approximately 720 iterations are required for the convergence in the parallel CG solver within each Newton's iteration.

Then, we use the mesh data of Table 3-5 to validate the solution of accuracy. Fig 3-9 shows the case for interaction between two like-charged spheres within a cylindrical pore (the same as case of Section 3-3) but use the second-order shape

function. The result shows that the solution of the second-order shape function will increase in accuracy over the first-order shape function. The inaccuracy reduces near to about 0.2%.

Chapter 4 Parallel Adaptive Mesh Refinement for Unstructured Tetrahedral Mesh

In this chapter, four major parts are described, including the basic algorithm for parallel adaptive mesh refinement (PAMR), coupling of PAMR with parallel Poisson-Boltzmann solver using Finite Element Method by a shell script of UNIX system, validation parallel performance of PAMR and application to a realistic three-dimensional problem. In each part, a general description and related specific algorithms are respectively provided.

4-1 Parallel Adaptive Mesh Refinement

4-1-1 Basic Algorithm of Parallel Adaptive Mesh Refinement

In the adaptive mesh refinement module, the data to record new added nodes is based on cell. In other words, every cell would only know new added nodes that belong to it. The number of added nodes is fixed in each process. However, before to divide cells, the number of added nodes must be unique in all processors. Besides, in this module, it needs a neighbor identifying arrays. This array defines the interfacial cell for each face. It would record the global cell number of interfacial cells for each face of the cell.

At each mesh refinement step, individual edges are marked for refinement, or no

change, based on an error indicator calculated from the solution, for example, the electric field. These cells which need to refine would add new nodes on each edge. They are called isotropic cells. For two-dimensional mesh, a parent cell [Lian *et. al.*, 2006] is divided to form four child cells. For three-dimensional mesh and tetrahedral cell, the parent cell is divided to form eight child cells, as show in Fig 4-1.

When isotropic cells add nodes on their edges, the cells neighbor them would appear one to three handing nodes at the same time. The cells which have hanging nodes but not belong to isotropic cells are called anisotropic cells. In order to remove hanging nodes in anisotropic cells, it has some procedures to do. Using two-dimensional mesh as an example is shown in Fig 4-2.

Then, we rewrite the above that considering the complexity of involved logic and data structure, we are only interested in refining the unstructured tetrahedral mesh. The general idea of refining the mesh in the current study is to simply “isotropically” refine the parent cells that require refinement into eight child cells. These child cells from isotropic refinement are classified as “isotropic cells”. In contrast, those cells other than the isotropic cells are then classified as “anisotropic cells”. The important steps in refining cells can be summarized as follows. (1) Add a node on the cell edges of “isotropic” and “anisotropic” cells. (2) Refine cells that have added node(s) on the edge(s), and (3) update connectivity and cell neighbor-identifying information.

When hanging nodes appear in a triangular cell, the cell must be divided into different way. However, for three-dimensional mesh, the division is more complex. The division would consider the number and position of the hanging nodes, to decide whether to add new nodes to obey the refinement rules. There are three results. One is to divide into eight child cells. Another is to divide into four child cells. The other is to divide into two child cells. The detail refinement rules are be show in Fig 4-3 [Wu, 2002].

4-1-2 Cell Neighboring Connectivity

In general, the AMR scheme [MacNeice *et. al*., 2000; Norton *et. al*, 2001; Oliker *et.al.*, 2000] often applies the edge-based data structure in storing and searching connectivity-related data. With this edge-based data structure, the cells containing a common refined edge can be easily identified, which is necessary in the AMR scheme. However, the edge-based data structure is relatively *memory demanding*. For example, up to ~7N edges can result from a tetrahedral mesh with N nodes, while only ~6N cells are formed. With this edge-based data structure, ~10 times of the number of edges are often required since up to ~10 or more cells share a single edge in a typical tetrahedral mesh, while only four times of the number of the cells are required (four faces in a cell) if the cell-based data structure is employed. Thus, the overall memory saved in this regard can be up to 2~3 times if the cell-based data structure is utilized. This justifies the use of the cell neighbor-identifying array in the present study.

Since we do not utilize the edge-based data structure, how to quickly search the cells sharing a common edge becomes very important in the refinement procedure. The basic concept of the cell-based data structure is based on the fact that only two faces share a common edge in a cell, which can be used to efficiently sort the sharing faces. Fig. 4-4a shows a typical example of the common edge shared by the adjacent cells. Edge A-B is shared by Cell A-B-1-2, A-B-2-3, A-B-3-4, A-B-4-5, and A-B-5-1. For brevity, the procedure of sorting the faces sharing the common Edge A-B is used as an example to demonstrate the general ideas of face sorting using cell-based data structure and is summarized as follows:

1 Define Cell A-B-1-2 as the initial searching cell.

2 Search the neighboring cells (A-B-2-3 and A-B-5-1) which share the common edge (Edge A-B) according to the cell neighboring-identifying array and finite element connectivity.

3 Select one of these two neighboring cells (e.g., Cell A-B-2-3 through Cell Interface A-B-2 shown in Fig. 4-4b) as the current searching cell.

4 Mark Edge A-B as a refined edge with local edge numbering in Cell A-B-2-3.

5 Search the next neighboring cell (Cell A-B-3-4 through Cell Interface A-B-3) which shares the common edge (Edge A-B) with Cell A-B-2-3 according to the cell neighboring-identifying array and finite element connectivity.

6 Assign the next neighboring cell as the current searching cell.

7 Repeat the similar process in Procedures 4-5 until the next neighboring cell is the same as the initial searching cell defined in Procedure 1. For example, as shown in Fig. 4-4c, the neighboring cell of face i' of Cell A-B-1-5 (the same as the face j' of Cell A-B-1-2) is the Cell A-B-1-2, which is the defined initial searching cell.

8 For cases in which this cell-by-cell searching process meets the physical face boundary or IPB (Interior Processor Boundary), the other neighboring cell (e.g., A-B-5-1 through Cell Interface A-B-1) of the initial searching cell (Cell A-B-1-2) will be selected as the current searching cell. The algorithm will repeat the similar process in Procedures 4-5 until another physical face boundary or IPB is found in the process of identifying the cell group.

4-1-3 Cell-Quality Controls

In the process of adaptive mesh refinement, a refined mesh which has a larger aspect ratio often appears in the final mesh distribution, especially in the "anisotropic" cells. Most mesh-smoothing schemes tend to change the structure of a given mesh to achieve the "smoothing effect" (a better aspect ratio) by rearranging the position of the nodes in the mesh. Changes made by a smoothing scheme, however, could

possibly modify the desired distribution of mesh density produced by the AMR procedure. Additionally, the cost of performing a global mesh smoothing could be high. Alternatively and in a simpler way, it is possible to prevent or slow down the degradation of cell quality during a repeated adaptive refinement process. The present cell-quality control scheme classifies the cells based on how they will be refined. This allows us to avoid creating cells with large aspect ratios during the refinement. After identifying those cells, we can then refine them with a better refining strategy.

The basic idea of the proposed mesh-refining scheme is to simply prevent the further deterioration of cell quality while refining the cells that originate from previously "anisotropic" cells. The proposed procedures of the cell-quality control scheme are shown schematically in Fig. 4-5 [Jeng, 2004] along with the corresponding procedures without cell-quality control. The logic looks quite complicated at first glance; however, it is conceptually simple in reality. A general rule of thumb is to identify if the hanging nodes occur at the "normal edge" or "short edge" of the cell. Note that the "normal edge" is defined as one of the six edges of an isotropic cell, while the "short edge" results from the bisection of a "normal edge" in a previous level of refinement. If at least one of the hanging nodes occurs at the short edge, special treatment such as that in Fig. 4-5 [Jeng, 2004] is required to prevent the further deterioration of cell quality in later mesh refinement. Otherwise, the procedure

in Fig. 4-3 [Wu, 2002] applies directly.

In the data structure, we need to record each previously refined cell with an index showing if it is an “isotropic” cell or an “anisotropic” cell (two or four refined cells, respectively). This can then be used for cell-quality control purposes as shown in Figs. 4-5a and 4-5b. The latter figures show the corresponding process of refining the cell if the parent cell originates from an “anisotropic” cell due to two or four refined cells, respectively.

For example, Fig4-6 shows a typical cell (1-2-3-4). Because it is affecting by other cell, it has hanging nodes (8, 9, A). In order to handle hanging nodes, we connect nodes (8, 9, A) to node 4. But the aspect ratios of typical cells (1-4-8-A, 5-4-8-9 and 7-4-9-A) will be worse. However, if we add three nodes (B, C and D), and connect them with other nodes. Those eight child typical cells (4-B-C-D, 1-8-A-B, 5-8-9-D, 7-9-A-C, 8-9-A-B, 8-9-B-D, 9-A-B-C and 9-B-C-D) would have better aspect ratios.

However, it often creates too many isotropic child cells, which tremendously increases the computational load of the solver. For example, in the present implementation as shown in Fig. 4-7, the original hanging node “8” is directly removed by adding two additional nodes (A and B) to form four child cells, instead of two child cells (1-4-8-7 and 5-4-8-7) from the parent cell 1-5-7-4. The present proposed

cell-quality control scheme produces refined cells with a better aspect ratio as compared to the "isotropic" refinement presented [Wu *et. al.*, 2004]. So cell-quality-control will allow this division and not to affect it. And, it would get better aspect ratios.

If the boundaries of the computational domain are not straight, it is not sufficient to place the new node in the midway of the edge of the face of the parent cell. If this is done, it would dual to a rough piecewise representation of the original geometry results. What must be done is to move the new node location onto the real boundary contour surface. In the current implementation, it is assumed that the boundaries can be represented in parametric format. Specific neighbor identifiers are assigned to these non-straight boundary cells to distinguish from straight boundary cells. Whenever the boundary cells, which require mesh refinement, are identified as a non-straight boundary cell, the corresponding parametric function representing the surface contour are called in for mesh refinement to locate the correct node positions along the parametric surface.

4-1-4 Surface Cell Refinement

If the boundary surface of the computational domain is not planar, it is not sufficient to place the added nodes directly on the edge (called "boundary edge"

hereafter) of the parent-cell face that belongs to the boundary. Should the refined nodes be placed directly on the boundary edges, this may result in the rough piecewise representation of the original boundary surface. What must be done is to move the position of the added nodes onto the real boundary contour surface. In the present implementation, it is assumed that the boundary surface can be represented in a parametric format (a second-order polynomial) in several segments specified by the user. Special cell neighbor identifiers are assigned to these curved boundary cells to distinguish them from straight boundary cells within the computational domain. Whenever the boundary cell which requires refinement is identified as a curved boundary cell, the corresponding parametric function representing the true surface contour is called in to locate the correct node positions along the parametric surface.

4-1-5 Modules of Parallel Adaptive Mesh Refinement

All modules in the core of the PAMR are explained in detail as follows [Jeng, 2004]:

Module I. Add nodes on cell edges:

I-1. Add new nodes on all edges of "isotropic" cells that require refinement. These cells are called "isotropic" cells since it is refined from one cell to eight smaller cells.

I-2. Add new nodes to “anisotropic” cells, which may require further treatment in the following steps, if hanging nodes exits in these cells... Note that these cells are called “anisotropic” since they are not those cells originally require “isotropic” mesh refinement.

I-3. Communicate the hanging-node data to corresponding neighboring processors if the hanging nodes are located at the IPB (Interface Processor Boundary). If hanging-node data are received from other processors in this stage, then go back to Step I-2 for updating the node data. If there is no hanging-node data received at this stage, then move on to the next step.I-4. Remove hanging nodes based on refinement rules, if need to add new nodes go to I-2. If all cells obey the rules without to affect them, it mean all new nodes are be added.

I-4. Remove hanging nodes based on Hanging-node Removal rules. Go to Step I-2 to add new nodes if some more nodes addition is required. If there is no more node addition required, then ends this module execution.

Module II. Renumber added nodes:

II-1. Add up the number of the new added nodes in each processor, excluding those on the IPB.

II-2. Communicate this number to all processors. These are used to add up the

total number of added nodes in the interior of all processors.

II-3. Renumber those added nodes in the interior of the processor according to results from Step-II-2.

II-4. Communicate data of added nodes on IPBs among all processors.

II-5. Renumber the added nodes on IPBs on all processors.

Module III. Update connectivity data:

Form all new cells and define the new connectivity data for all cells.

Module IV. Build neighbor identifier array:

VI-1. Reset the neighbor identifier array.

VI-2. Rebuild the neighbor identifier array for all the cells based on the new connectivity data. Note that those nbr information on faces of the cells on the IPBs are not rebuilt and require further treatment in the next step.

VI-3. Record the neighbor identifier arrays that are not built in Step-IV-2, which the neighbor of the interested cell locates in the neighboring processor.

VI-4. Broadcast all the recorded data in Step-IV-3 to all processors.

VI-5. Build the nbr information on the IPBs, considering the overall connectivity data structure.

4-1-6 Procedures of Parallel Adaptive Mesh Refinement

Fig. 4-8 shows the proposed procedure of parallel mesh refinement for unstructured tetrahedral mesh. Only the general procedure is described in this report, while details of the parallel implementation can be found elsewhere [Lian *et. al*., 2006]. Basically, the parallel mesh refinement procedure in Fig. 4-8 is similar to those presented earlier for serial mesh refinement [Wu *et. al.*, 2004]. In serial mesh refinement, cells that require refinement are first identified and then refined "isotropically" into eight child cells. The generated hanging nodes are then removed following the procedure proposed in Wu *et al.* [2004] by further refining the cells including these hanging nodes into two, four, or eight child cells.

The detailed procedures and related data structure for parallel mesh refinement become much more complicated than those for serial mesh refinement, because parallel computing uses domain decomposition. Domain decomposition is also used in line with proposed parallelized of the Poisson-Boltzmann equation solver. Each spatial subdomain belongs to a specific processor in practice. Overall procedure, as shown in Fig. 4-8, can be summarized as follows:

1. Preprocess the input data at the host server and distribute them to all processors.

2. Index those cells that require refinement based on the refinement criteria. In the current study, a posteriori error estimator of the type proposed by Zienkiewicz and Zhu [1987] for cell refinement is used.

3. Check if further mesh refinement is necessary. If yes, then proceed to the next step. If no, then proceed to **Step 9**.

4. Add new nodes into those cells that required refinement.

   4a. Add new nodes onto all edges of isotropic cells.

   4b. Add new nodes into anisotropic cells that require further refinement as decided in the following steps.

   4c. Communicate the hanging-node data to corresponding neighboring processor if the hanging nodes are located at inter-processor boundary (IPB).

   4d. Remove hanging nodes following the procedure as shown in Wu, *et al.* [2004]. The basic idea is to remove hanging nodes for all kinds of conditions and refine each indexed cell further into 2, 4, or 8 child cells.

5. Unify the global node and cell numberings due to the newly added nodes among all processors,

   5a. Add up the number of newly added nodes in each processor, excluding

those located at IPBs.

5b. Gather this number from all other processors and add them up to obtain the updated total number of nodes, including old and new nodes, but excluding the newly added nodes at IPBs.

5c. Build up the updated node-mapping and corresponding cell-mapping arrays for those newly added nodes in the interior part of each subdomain based on the results in **Step 5b**.

5d. Communicate data of newly added nodes at IPBs among all processors.

5e. Build up the node-mapping array for the received new nodes at IPBs in each processor.

6. Build up new connectivity data for all cells due to the newly added nodes.

7. Build up the new neighbor-identifying array based on the new connectivity data obtained in **Step 6**.

7a. Reset the neighbor-identifying array.

7b. Build up the neighbor-identifying arrays for all cells based on the new connectivity data, excluding the data associated with the faces lying on the IPBs that require the updated information of the global cell number, which is not known at this stage.

7c. Record all the neighbor-identifying arrays that are not rebuilt in **Step 7b**.

7d. Broadcast all recorded data to all processors.

7e. Build up the neighbor-identifying arrays on the IPBs, considering the overall connectivity data structure.

8. Decide if it reaches the preset maximum number of refinement. If it does, then proceed to the next step; otherwise return to **Step 3**.

9. Synchronize all processors.

10. Host gathers all data and outputs the data.

4-2 Coupling of PAMR with Parallelized Poisson-Boltzmann Equation Solver

4-2-1 *A posteriori* error estimator

An *a posteriori* error estimator of the type proposed by Zienkiewicz and Zhu [1987] is used for estimating the error of the solution.

In the case of Poisson-Boltzmann Equation, the Finite Element solution $\hat{E}$ for the electric field, which is, to within a sign, a gradient of electrostatic potential, is compared with an alternative postprocessed recovered solution. The latter is called the "exact" solution and will be designated $\widetilde{E}$. To obtain $\widetilde{E}$, a very simple gradient recovery procedure based on averaging the nodal values of the Finite Element solution $\hat{E}$ is used. The components of $\widetilde{E}$ are expressed via the same basic functions that were

used for the electrostatic potential and are continuous.

In the case of the exact solution variables $\hat{E}$ and $\widetilde{E}$ must coincide. In general, the error is defined as a deviation of one variable from the other. More precisely, the global absolute error is defined by expression

$$\delta = \left\{ \int_{\Omega} [(\hat{\mathrm{E}}_{\mathrm{x}} - \widetilde{\mathrm{E}}_{\mathrm{x}})^2 + (\hat{\mathrm{E}}_{\mathrm{y}} - \widetilde{\mathrm{E}}_{\mathrm{y}})^2 + (\hat{\mathrm{E}}_{\mathrm{z}} - \widetilde{\mathrm{E}}_{\mathrm{z}})^2] \mathrm{dxdydz} \right\}^{1/2} \tag{4-1}$$

The integral begin taken over the whole domain Ω. This is simply a L^2-norm of the difference $\hat{E} - \widetilde{E}$. The global relative error ε is defined by

$$\varepsilon = \frac{\delta}{\left\| \hat{E} \right\|} = \frac{\delta}{\left\{ \int_{\Omega} (\hat{E}_x^2 + \hat{E}_y^2 + \hat{E}_z^2) dxdydz \right\}^{1/2}} \tag{4-2}$$

Absolute error δ_i in a particular element i is defined as in Equation (4-1) except that the integral is taken over the element. The relation

$$\delta^2 = \sum_{i=1}^{N} \delta_i^2 \tag{4-3}$$

must hold, where N is the total number of elements in the mesh.

The level of accuracy is determined by the prescribed global relative error ε_{pre} and the corresponding absolute error $\delta_{pre} = \varepsilon_{pre} \left\| \hat{E} \right\|$. Given δ_{pre}, the current mean absolute error δ_m in an element can be calculated according to

$$\delta_m = \left[\frac{\delta_{pre}^2}{N} \right]^{1/2} = \left[\frac{(\varepsilon_{pre} \left\| \hat{E} \right\|)^2}{N} \right]^{1/2} \tag{4-4}$$

The current number of elements would each have to have this value of absolute

error to provide the prescribed level of accuracy, on the condition that absolute errors in all the elements were the same. Those elements whose absolute errors exceed the current mean absolute error δ_m are chosen to be subjected to subdivision.

All mesh need some means to detect the requirement of local mesh refinement to better capture the variations EDL fields and hence to obtain more accurate numerical solutions. This also applies to parallel Poisson-Boltzmann solver. It is important for the refinement parameters to detect a variety of EDL. In literature, there are many criteria refinement. The simple criterion is the electric field usually to be adopted as the refinement parameter. The electric field is the potential gradient. Beside, *a posteriori* error estimator of the type proposed by Zienkiewicz and Zhu [1987] is common criterion of refinement. A postprocessed recovered "exact" solution of the electric field can be obtained from various gradient recovery schemes applied to the FE solution of electric field. The FE solution of the electric field is then compared with this "exact" solution of the electric field. A very simple gradient recovery scheme based on the averaging of cell values, of the FE solution of the electric field is employed. In the present study, a prescribed global relative error ε_{pre} of 0.0003 is used to control the level of accuracy.

4-2-2 Parallelized Poisson-Boltzmann Equation Solver with PAMR

The PAMR presented in the previous section can be easily coupled to the current parallel Poisson-Boltzmann equation solver since it uses 3D unstructured tetrahedral mesh and MPI for data communication. The PAMR can therefore be easily packaged as a library and inserted into the source code of any parallel numerical solver. However, memory conflicts between the inserted library and the numerical solver itself may occur and result in a reduction of the problem size that one can handle in practice. To overcome this, a simple coupling procedure written in shell script (Fig. 4-9), which is standard on all Unix-like systems, was prepared to link the PAMR and the current parallel Poisson-Boltzmann equation solver. In doing so, we kept the application source codes intact without making any changes to it. This is especially justified if only the steady state of the physical problem is sought, in which normally only a few mesh refinements are necessary for a fairly satisfactory resolution to the problem. Thus, the total I/O time, in proportion to the number of couplings, in switching between two codes can be reduced to a minimum in practical applications. In addition, as shown in Fig. 4-9, after identifying those cells that require refinement, the domain is repartitioned using PMETIS which balances the workload among the processors based on the approximate final number of nodes in each processor as the weighting factor.

4-3 Validation and Parallel Performance of PAMR

4-3-1 Validation of the PAMR

To test the PAMR using the PPBES, consider a test case with two free identical charged spheres (radius=5), a sphere radius to pore radius ratio of λ=0.416 and a nondimensional separation distance of 0.5. A surface potential of Ψ_0=2.0 across the spheres is used to demonstrate the advantage of applying adaptive mesh refinement. Due to symmetry, only 1/16 of the whole domain is simulated. In the present study, *a posteriori* error estimator of the type proposed by Zienkiewicz and Zhu [1987] and implemented by Dyshlovenko [2001] for axisymmetric 2D Poisson-Boltzmann equation is used for estimating the solution error during the mesh-refining process. A postprocessed recovered "exact" solution of the electric field can be obtained from various gradient recovery schemes applied to the FE solution of electric field. The FE solution of the electric field is then compared with this "exact" solution of the electric field. A very simple gradient recovery scheme based on the averaging of cell values, instead of nodal values as used in Ref.[Zienkiewicz and Zhu, 1987], of the FE solution of the electric field is employed. This is attributed to the fact that the electric field at the nodes are discontinuous since we only utilize a linear shape function in the current study, rather than a quadratic shape function as used in Ref.[Zienkiewicz and Zhu, 1987]. In the present study, a prescribed global relative error ε_{pre} of 0.0003 is used to

control the level of accuracy. The absolute error in each element is then compared with the current average absolute error based on the ε_{pre} at each level to decide if refinement is required. In addition, the method for calculating interaction force using the potential distribution on the sphere surfaces is described in detail elsewhere, such as Dyshlovenko [2001], and is skipped here for brevity.

Table 4-1 using first-order shape function and Table 4-2 using second-order shape function summarize the evolution of the number of elements, the number of nodes, and the computed force of interaction between two free spheres at different refinement levels for first-order and second-order shape function. Note that the force of interaction is obtained by integrating the simulated electric field at the spherical surface as in previous studies [Dyshlovenko, 2001]. In Table 4-1 (the first-order shape function), The resulting number of nodes increased from an initial of 924 to a final (level-5) of 36,316; while the strength of the interaction force increased from an initial value of 20.9924 to a final value (level-5) of 48.4012, which is very close to the value of 48.8360 obtained by Dyshlovenko [2001] using an axisymmetric code. However, when we use the second-order shape function (Table 4-2), a value of final mesh level (level-6) of 48.8242, which is very close to the value of 48.8360 obtained by Dyshlovenko [2001], there are only difference 0.04%. Fig. 4-10a and Fig. 4-10b show the corresponding surface mesh distribution for first-order shape function at the

initial and final (level-5) refinement levels, respectively. The mesh quality does not deteriorate much after five levels of refinement using the present PAMR. All the simulations shown in later sections apply this adaptive mesh-refining technique, unless otherwise specified.

4-3-2 Parallel Performance of the PAMR

The parallel performance of the PAMR is studied using a refined mesh generated during the process of AMR (after 3 AMRs) using the DSMC method. Before the fourth PAMR, the original mesh (level-3) has 1,070,194 tetrahedral cells and 187,649 nodes (Fig. 4-11a), while after the fourth PAMR, the amount increases to 2,701,178 cells and 468,944 nodes (Fig. 4-11b). A test using this fairly large size of mesh lies mainly on the fact that the current PAMR is designed to handle a large mesh size.

Resulting timings (in seconds) for different components of the PAMR using different numbers of processors (up to 64), respectively, are listed in Table 4-3 and Fig 4-12. It is clearly shown that the timing for the preprocessing module increases slightly with the increasing number of processors, although the increase seems to level off as the processor number ≥ 32. This mild increase can be attributed to the initial data distribution from the host processor to the increasing number of slave processors. The advantages of the parallel implementation of mesh refinement can be clearly seen from modules I, II, and IV, of which the timings reduce dramatically with the increasing number of processors. For example, as the number of processors increases from 2 to 32, the timing for module I, II, and IV reduces to approximately 61.4, 95.9, and 9.3 times, respectively, while it increases to 2.4 times for module III.

The corresponding overall computational time decreases 29.7 times (from 1103s to 37.1s). Nevertheless, the timing levels off as the number of processors increases up to 64 due to the increase of communication among processors. In addition, the total computational time decreases even more to 62 times (from 1090.5s to 17.6s) if the preprocessing time is excluded. From our experience, reading the grid data (Section 4-1-6 Procedures of PAMR, **Step 1**) takes most of the time in the preprocessing module. Thus, the preprocessing time can be greatly reduced if the parallel processing of **Step 1** is implemented, although we have not done so in the present study. The resulting increase of computational speed, excluding the preprocessing, is approximately scaled as $N^{1.5}$ for $N_{proc} \leq 32$ which is surprisingly good. This high speedup is obviously due to the dramatic decrease of the time spent in Module II, which is the most time-consuming part of the mesh-refining algorithm. The reasons for this rapid decrease of processing time in Module II (Section 4-1-6 Procedures of PAMR, **Steps 5a-5e**) include a possible cache miss of the superscalar machine due to the large problem size and a highly localized algorithm in renumbering the added nodes, except in **Steps 5b** and **5d** which require moderate communication among processors.

4-4 Application to a Realistic Three-dimensional Problem

A completed parallelized Poisson-Boltzmann equation solver with parallel adaptive mesh refinement using *a posteriori* error estimator is used to compute the force of interaction between two identical charged spheres (radius a=0.325μm, κa =1.185, and surface potential Ψ_0=3.0) near a charged infinite flat plate (surface potential Ψ_0=5.0) as shown schematically in Fig. 4-13. Note that this is a well-known case that was

experimentally studied previously [Larsen and Grier, 1997]. Distance h from the flat plate to the sphere centers are fixed at 2.5μm ($a/h = 0.13$), while the separation distance r between the spheres are varied to study its influence on the interaction force F. This example only serves to demonstrate the capability of the present parallelized Poisson-Boltzmann equation solver using FEM with PAMR in predicting the electrostatic potential distribution with three-dimensional complicated geometries. No thorough physical interpretation or parametric study was explored in the current study, but these are of course worthy of further investigation.

Fig. 4-14a and Fig. 4-14b, illustrate the typical initial and final (level-6) surface mesh distributions (h=2.5μm, r=3μm), respectively, while Table 4-4 summarizes the corresponding evolution of the interaction force between the spheres, along with the number of nodes and elements for first-order shape function. The resulting number of nodes increases from ~0.016 to ~0.186 million for this typical case, while the force of interaction increases from 5.432e-16 Newton to 8.325e-15 Newton, which is very close to the experimental measurement [Larsen and Grier, 1997]. Fig. 4-15a and Fig. 4-15b illustrate the distribution of electrostatic potential around the charged sphere for h=2.5μm ($a/h = 0.13$) and h=9.5μm ($a/h = 0.03421$), respectively, and also show the complicated interaction between the two spheres when the flat plate is near the spheres (r=3.0μm).

Besides, Table 4-5 and Table 4-6 summarize the corresponding evolution of the interaction force between the spheres, along with the number of nodes and elements for the first-order and the second-order shape function (h=2.5μm, r=4.5μm).

Finally, Fig. 4-16 presents the computed force of interaction as a function of distance r at h=2.5μm, along with fitted experimental measurements by Larsen and Grier [1997] show that the results of using the first-order and second-order shape function. Note that the experimental fitting of the interactive force is obtained by taking the derivative of the fitted measured potential data. Computational results agree well with the experimental data within experimental uncertainties. In particular, the attractive force (negative F) at some range of separation distance r was predicted successfully, which to the best of the authors' knowledge has never been reproduced by solving directly the Poisson-Boltzmann equation in the literature, although one simulation report using Brownian dynamics simulations [Squires and Brenner, 2000] does exist.

Chapter 5 Concluding Remarks

5-1 Summary

In this thesis, a parallelized 3-D nonlinear Poisson-Boltzmann equation solver (PPBES) using finite element method (FEM) with parallel adaptive mesh refinement (PAMR) is completed and verified. The results of research are described as follows.

In the first phase, the parallelized 3-D nonlinear Poisson-Boltzmann equation solver is completed using Galerkin finite element method with unstructured tetrahedral mesh. Major findings of this phase are summarized as:

1. Interpolation within a typical element includes the first-order and second-order shape functions. The second-order shape function performs much better than the first-order shape function in solution accuracy for comparable runtime and memory.
2. Inexact Newton iterative scheme is very effective for convergence of PPBES within 3-5 iterative steps depending upon the initial guess.
3. The Jacobian matrix for Newton iterative scheme is simplified as a diagonal matrix using nodal quadrature for the integration of the source term, which both reduces the computational load of the Jacobian matrix and simplifies the parallel implementation.
4. Completed PPBES is validated using several quasi-2-D cases by comparing

excellently either with previous analytical, approximate or numerical solutions.

5. The parallel performance is studied on a HP PC-cluster system at NCHC using close to 0.1 million nodes and 0.61 million elements. Results show that 76.2% of parallel efficiency can be reached at processors of 32, which can greatly reduce the runtime for realistic 3-D applications.

In the second phase, an *h*-refinement based PAMR scheme for an unstructured tetrahedral mesh using dynamic domain decomposition on a memory-distributed machine is developed and tested in detail. Then, the coupled PPBES-PAMR code using *a posteriori* error estimator is used to simulate a realistic three-dimensional problem. Major findings in this phase are summarized as:

1. PAMR procedures include isotropic refinement from one parent cell into eight child cells and then followed by anisotropic refinement, which can effectively remove the hanging nodes, with a simple mesh-quality control scheme.

2. Parallel performance of this PAMR is studied on a PC-cluster system up to 64 processors using approximately one million tetrahedral cells. Results show that the parallel speedup scales approximately as $N^{1.5}$ for $N_{proc} \le 32$ which is surprisingly good, where N is the number of processors.

3. Completed PAMR code using the PPBES is used to simulate two like-charged spheres in a cylindrical pore. Results show that PAMR can systematically increase the solution accuracy of the PPBES. The value of final mesh level of 48.4012 using the first-order shape function, which is very close to the value of 48.8360 obtained by Dyshlovenko [2002] using an axisymmetric code. However, as we use the second-order shape function, a value of 48.8242 using the final refined mesh is obtained which is only 0.04% with previous numerical value with Dyshlovenko [2002] using a 2-D axisymmetric PB equation solver.
4. Finally, the coupled PPBES-PAMR code is used to simulate the interactive force between two like-charged spheres near a same charged planar wall. Results are in excellent agreement with experimental data considering the experimental uncertainties, which is the first simulation in the literature to the best knowledge of the author.

In the present study, we have completed a coupled PPBES-PAMR code that is very accurate and efficient for future simulations of realistic problems in colloidal systems.

5-2 Recommendations for Future Work

Based on the study in this thesis, recommendations for future work are summarized as follows:

1. To improve the efficiency of the parallel matrix solver by using a robust preconditioning technique such as additive Swartz method.
2. To extend the PPBES into a code that can handle hybrid mesh. We have developed a code of hybrid mesh (using the first-order and the second-order shape function) for single CPU. Figure 5-1 shows the potential profiles that compare the code of tetrahedral mesh with code of hybrid mesh of collapsed nodes which the maximum inaccuracy 0.2%. The test case is the distribution of potential around the charged spheres near a like-charged plate for mesh of level 0. The code could be developed as the parallel version coupled with PAMR in the future. It will improve the solution of accuracy after refinement. For more information, discuss the related skills in Appendix B.
3. To utilize the completed coupled PPBES-PAMR code to simulate particle interaction in several important colloidal related problems. These results can provide important information to researches in bio-chemistry related field.

REFERENCES

1. Almasi, G. and Gottlieb, A., “Highly Parallel Computing,” Benjamin/Cummings, Red-Wood city, CA, 2nd Editon 1989.
2. Bowen, W.R. and Sharif, A.O., “Long-range electrostatic attraction between like-charge spheres in a charged pore,” Nature, Vol.393, pp.663-665, 1998.
3. Bowen, W.R. and Sharif, A.O., “Adaptive finite-element solution of the nonlinear Poisson-Boltzmann equation: A. charged spherical particle at various distances from a charged cylindrical pore in a charged planar surface,” Journal Colloid Interface Sci., Vol.187, pp.363-374, 1997.
4. Brackbill, JU, “An adaptive grid with direction control,” Journal of Computational Physics, Vol.108, pp.38–50, 1993.
5. Baker, N., Holst, M. and Wang, F., “Adaptive multilevel finite element solution of the Poisson-Boltzmann equation II. Refinement at solvent-accessible surfaces in biomolecular systems,” Journal of Computational Chemistry, Vol.21, pp. 1343-1352, 2000.
6. Baker, N. A., Sept, D., Joseph, S., Holst, M. J. and McCammon, J. A., “Electrostatics of nanosystems: Application to microtubules and the ribosome,” Proceedings of the National Academy of Sciences of the United States of America, 98, pp.10037-10041, 2001a.

7. Baker, N., Sept, D., Holst, M. and McCammon, J. A.,"The adaptive multilevel finite element solution of the Poisson-Boltzmann Equation on massively parallel computers," IBM Journal of Research and Development, Vol.45, No.3/4, page 427, 2001b.

8. Balls, G. T. and Colella, P., “A Finite Difference Domain Decomposition Method Using Local Corrections for the Solution of Poisson's Equation, Journal of Computational Physics, Vol.180, pp.25-53, 2002.

9. Bank, R. E. and Holst, M., “A new paradigm for parallel adaptive meshing algorithms,” Siam Journal on Scientific Computing, Vol.22, pp.1411-1443, 2000.

10. Carnie, S. L., Chen, D. Y. C., and Stankovich, J., "Computation of forces between spherical colloidal particles: Nonlinear Poisson-Boltzmann Theory," Journal of Colloid and Interface Science, Vol.165, pp.116-128, 1994.

11. Cortis, C. M. and Friesner, R. A., “An automatic three-dimensional finite element mesh generation system for the Poisson-Boltzmann equation,” Journal of Computational Chemistry, Vol.18, pp.1570-1590, 1997a.

12. Cortis, C. M. and Friesner, R. A., “Numerical solution of the Poisson-Boltzmann equation using tetrahedral finite-element meshes,” Journal of Computational Chemistry, Vol.18, pp.1591-1608, 1997b.

13. Collins, John and Lee, Abraham P., “Microfluidic flow transducer based on the

measurement of electrical admittance," Lab on a Chip, NO.4(1), pp.7-10, 2004.

14. Connell, S.D. and Holms, D.G., "Three-Dimensional Unstructured Adaptive Multigrid Scheme for the Euler Equations," AIAA Journal, Vol.32**,** pp.1626-1632, 1994.
15. Chandra, R., Dagum, L., Kohr, D., Maydan, D., J. McDonald, "Parallel programming in OpenMP," Morgan Kaufmann Publishers, San Francisco, CA, 2000.
16. Das, P.K. and Bhattacharjee, S., "Finite Element Estimation of Electrostatic Double Layer Interaction between Colloidal Particles inside a Rough Cylindrical Capillary: Effect of Charging Behavior," Colloids and Surface A, Vol.256, pp.91-103, 2005.
17. Dyshlovenko, Pavel, "Adaptive Mesh Enrichment for the Poisson-Boltzmann Equation," Journal of Computation Physics, Vol.172, pp.198-208, 2001.
18. Dyshlovenko, Pavel, "Adaptive numerical method for Poisson-Boltzmann equation and its application," Computer Physics Communications, Vol.147, pp.335-338, 2002.
19. Fogolari, Federico, Zuccato, Pierfrancesco, Esposito Gennaro and Viglino, Paolo, "Biomolecular Electrostatics with the Linearized Poisson-Boltzmann Equation," Biophysical Journal, Vol.76, pp.1-16, 1999.

20. Gilson, M. K., Davis, M. E., Luty, B. A. and McCammon, J. A., "Computation of electrostatic forces on solvated molecules using the Poisson-Boltzmann equation," Journal of Physical Chemistry, Vol.97, pp.3591–3600, 1993.

21. Gowda, Shivaraju B. “A Comparison of Sparse & Element-by-Element Storage Schemes on The Efficiency of Parallel Conjugate Gradient Iterative Methods for Finite Element Analysis”, MS thesis, Graduate School of Clemeson University, 2002.

22. Hunter, R. J., “Foundations of Colloid Science,” Vol.2, Oxford: Clarendon Press, 1989.

23. Holst, M., Baker, N. and Wang ,F., “Adaptive multilevel finite element solution of the Poisson-Boltzmann equation I. Algorithms and examples, Journal of Computational Chemistry, Vol.21, pp.1319-1342, 2000.

24. Holst, M., Kozack, R. E., Saied, F. and Subramaniam, S., “Protein electrostatics: rapid multigrid-based Newton algorithm for solution of the full nonlinear Poisson-Boltzmann equation,” Journal of Biomolecular Structure and Dynamics, Vol.11, pp.1437-45, 1994a.

25. Holst, M., Kozack, R. E., Saied, F. and Subramaniam, S., “Treatment of electrostatic effects in proteins: multigrid-based Newton iterative method for solution of the full nonlinear Poisson-Boltzmann equation,” Proteins, Vol.18, pp.

231-45, 1994b.

26. Hoskin, N. E., "The interaction of two identical spherical colloidal particles I--Potential Distribution," Proceedings of the Royal Society (London), Series A, 248, pp.433-448, 1956.

27. Hestenes, M. and Stiefel E. ,"Methods of Conjugate Gradient for Solving Linear Systems," Journal of research of the National Bureau of Standards, Vol.49, pp.409-439, 1952.

28. Harries, Daniel, "Solving the Poisson-Boltzmann Equation for Two Parallel Cylinders," Langmuir, Vol.14, pp.3149-3152, 1998.

29. Wu, Fu-Yuan, "The Three-Dimensional Direct Simulation Monte Carlo Method Using Unstructured Adaptive Mesh and It Applications," MS Thesis, NCTU, Hsinchu, Taiwan , July, 2002.

30. Hsu, K.-H., “Development of a Parallelized PIC-FEM Code Using a Three-Dimensional Unstructured Mesh and Its Applications,” PhD Thesis, NCTU, Hsinchu, Taiwan, July, 2006.

31. Kuo, Chia-Hao, "The Direct Simulation Monte Carlo Method Using Unstructured Adaptive Mesh and Its Applications," MS Thesis, NCTU, Hsinchu, Taiwan, June, 2000.

32. Kozack, R. E. and Subramaniam, S., “Brownian dynamics simulations of

molecular recognition in an antibody-antigen system," Protein Science, Vol.2, pp.915-926, 1993.

33. Kallinderis, Y. and Vijayan, P., "Adaptive Refinement-Coarsening Scheme for Three-Dimensional Unstructured Meshes", AIAA JOURNAL, Vol.31, No.8, 1993.

34. Karypis, G., Kumar, V., ParMETIS 3.1: An MPI-based Parallel Library for Partitioning Unstructured Graphs, Meshes, and Computing Fill-Reducing Orderings of Sparse matrices, 2003. Available from (http://www-users.cs.umn.edu/~karypis/metis/parmetis).

35. Larsen, A.E. and Grier, D.G., "Like-charge attractions in metastable colloidal crystallites," Nature, Vol.385, pp.230-233, 1997.

36. Lian, Y.-Y., Hsu, K.-H., Shao, Y.-L., Lee ,Y.-M., Jeng, Y.-W. and Wu, J.-S.,"Parallel Adaptive Mesh-Refining Scheme on Three-dimensional Unstructured Mesh and Its Applications," Computer Physics Communications (Accepted in May 2006).

37. Mackenzie, J. A. and Robertson, M. L., "A moving mesh method for the solution of the one-dimensional phase-field equations," Journal of Computational Physics, Vol.181, pp.526–544, 2002.

38. MacNeice, P., Olsonb, K.M., C. Mobarry, R. de Fainchtein, Packer, Charles,

Computer Physics Communications, Vol.126, pp.330, 2000.

39. MPI library, http://www-unix.mcs.anl.gov/mpi

40. METIS library, http://glaros.dtc.umn.edu/gkhome/views/metis

41. MANIFOLD CODE http://www.scicomp.ucsd.edu/~mholst/codes/mc

42. Nadeem, S A and Jimack, P K, “Parallel implementation of an optimal two level additive Schwarz preconditioner for the 3-D finite element solution of elliptic partial differential equations,” International Journal for Numerical Methods in Fluids, Vol.40, pp.1571, 2002.

43. Norton, C.D., Lou, J.Z. and Cwik, T., “Status and Directions for the PYRAMID Parallel Unstructured AMR Library,” Proceedings of the 15th International Parallel & Distributed Processing Symposium, IEEE Computer Society, Washington, DC, 2001.

44. Neu, J.C., "Wall-Mediated Forces between Like-Charged Bodies in an Electrolyte," Physical Review Letters, Vol.82, pp.1072-1074, 1999.

45. Oliker, L., Biswas, R., Gabow, H.N., “Parallel Tetrahedral Mesh Adaptation with Dynamic Load Balancing.,” Parallel Computing Journal, Vol.26, pp.1583-1608, 2000.

46. Okubo, T. and Aotani, S., “Microscopic observation of ordered colloids in sedimentation equilibrium and the importance of the Debye-screening length. 9.

Compressed crystals of giant colloidal spheres," Colloid & Polymer Science, Vol.266, No.11, pp.1042-1048, 1988.

47. Peano, A .G.., "Hierarchies of Conforming Finite Elements for Plane Elasticity and Plate Bending," Journal of Computers and Mathematics with Applications, Vol.2, pp.211-224, 1976.

48. Pao, C. V., "Block monotone iterative methods for numerical solutions of nonlinear elliptic equations," Numerische Mathematik, Vol.72, pp.239-262, 1995.

49. Prof. F.-N. Hwang, http://www.math.ncu.edu.tw/~hwangf

50. Quddus, N., Bhattacharjee, S. and Moussa, W., "An Electrostatic–Peristaltic Colloidal Micropump: A Finite Element Analysis," Journal of Computational and Theoretical Nanoscience, Vol.1, No.4, pp.438-444, 2004.

51. Rausch, R.D., Batina, J.T. and Yang, H.T.Y., "Spatial Adaption Procedures on Unstrutuctured Meshes for Accurate Unsteady Aerodynamics Flow Computation," AIAA Paper, No.91-1106-CP, 1991.

52. Russel, W. B., Saville, D. A., and Schowalter, W. R., "Colloidal Dispersions," Cambridge Univ. Press, Cambridge, England, 1989.

53. Shao, Z., Ren, C. L. and Schneider, G. E., "3D Electrokinetic Flow Structure of Solution Displacement in Microchannels for on-Chip Sample Preparation Applications", Journal of Micromechanics and Microengineering, Vol.16,

pp.589-600, 2006.

54. Sharp, K. A. and Honig, B., "Calculating total electrostatic energies with the nonlinear Poisson-Boltzmann equation," Journal of Physical Chemistry, Vol.94, pp.7684-7692, 1990.

55. Squires, Todd M. and Brenner, Michael P., “Like-Charge Attraction and Hydrodynamic Interaction,” Physical Review Letters, Vol.85, pp.4976-4979, 2000.

56. Saad, Yousef, “Iterative Method for Sparse Linear System,” Society for Industrial and Applied Mathematics, 2003.

57. Tuinier, R., “Approximate solutions to the Poisson-Boltzmann equation in spherical and cylindrical geometry,” Journal of Colloid and Interface Science, Vol.258, pp.45-49, 2003.

58. Thompson, Erik G., "Introduction to the Finite Element Method: Theory, Programming and Applications," John Wiley & Sons Inc, 2003.

59. Waltz, J., “Parallel adaptive refinement for unsteady flow calculations on 3D unstructured grids,” International Journal for Numerical Methods in Fluids, Vol.46, pp.37-57, 2004.

60. Wang, L. and Harvey, J.K., “The Application of Adaptive Unstructured Grid Technique to the Computation of Rarefied Hypersonic Flows Using the DSMC

Method," 19th International Symposium on Rarefied Gas Dynamics, Harvey J, Lord G (ed.), pp.843, 1994.

61. Wu, J.-S., Tseng, K.-C. and Wu, F.-Y., "Three Dimensional Direct Simulation Monte Carlo Method Using Unstructured Adaptive Mesh and Variable Time Step," Computer Physics Communications, Vol. 162, No. 3, pp.166-187, 2004.
62. Wu, Fu-Yuan, "The Three-Dimensional Direct Simulation Monte Carlo Method Using Unstructured Adaptive Mesh and It Applications," MS Thesis, NCTU, Hsinchu, Taiwan, July, 2002.
63. Yang, R.-J., Fu, L.-M. and Lin, Y.-C., "Electroosmotic Flow in Microchannels," Journal of Colloid and Interface Science, Vol.239, pp.98-105, 2001.
64. Zienkiewicz, O. C., J. P. de S. R. Gago and Kelly, D. W., "The Hierarchical Concept in Finite Element Analysis," Computers and Structures, Vol.16, No.1-4, pp.53-65, 1983.
65. Zienkiewicz, O.C. and Zhu, J.Z., "A Simple Error Estimator And Adaptive Procedure For Practical Engineering Analysis," International Journal for Numerical Methods in Engineering, Vol.24, pp.337-357, 1987.
66. Zienkiewicz, O.C. and Taylor, R.L., "The Finite Element Method," Butterworth-Heinemann, Oxford, 5th edition, 2000.

Newton iterations: the convergence criterion (ε_0=10^{-7}) (have a good initial guess)				
CG convergence criterion	CG=10e-7	CG=10e-6	CG=10e-5	CG=10e-4
CG Loop Iterations per Newton Iteration	~1300	~1140	~920	~870
Newton Loop Iterations	3	3	3	9
Initial Residual	1.19E-02	1.19E-02	1.19E-02	1.19E-02
Total Time(sec)	301.57	285.366	267.238	412.12

(a)

Newton iterations: the convergence criterion (ε_0=10^{-7}) (all zeros are used as the initial estimate)				
CG convergence criterion	CG=10e-7	CG=10e-6	CG=10e-5	CG=10e-4
CG Loop Iterations per Newton Iteration	~1300	~1140	~920	~870
Newton Loop Iterations	5	5	5	15
Initial Residual	8.68E-01	8.68E-01	8.68E-01	8.68E-01
Total Time(sec)	394.32	375.093	350.493	548.455

(b)

Table 3-1 The CG Solver and Newton Iteration for have a good initial guess and all zeros are used as the initial estimate. (CPU=6; 81,354 nodes; 458,064 elements).

Source	Debye-Huckel	Tuinier
Equation	$\Psi_{sp}(x) = \Psi_0\left(\frac{a}{a+x}\right)\exp(-x)$	$\Psi_{sp}(x) = \left(\frac{2a}{a+x}\right)\cdot \ln\left(\frac{1+t_0\exp(-x)}{1-t_0\exp(-x)}\right)$ $t_0 = \tanh\left(\frac{\Psi_0}{4}\right)$

Table 3-2 Analytical and approximate solutions for the simulation of the potential distribution around a charged sphere.

	1CPU	2CPU	4CPU	8CPU	16CPU	32CPU
Total time	2486.63	1681.31	744.86	341.96	161.01	102.19
CG time	1243.33	857.46	387.93	181.24	86.94	56.20
Setup matrix time	220.37	143.45	67.33	34.56	20.23	12.95
Impose BC time	8.32E-2	3.32E-2	8.43E-3	4.73E-3	2.42E-3	1.23E-3
Setup Shape Function. time	0.42	0.23	7.32E-2	4.31E-2	2.68E-2	8.52E-3
Index FEM time	---	6.14E-2	2.35E-2	2.34E-2	4.41E-2	6.41E-3
Send/Receive time	---	14.24	15.43	16.32	16.54	17.53
All reduce time	---	3.03	6.56	6.94	8.52	11.33
Others time	1022.41	662.81	267.43	102.82	28.71	4.16

Table 3-3 The time breakdown (in seconds) for various components of the parallelized Poisson-Boltzmann equations solver (electrostatic potential distribution around a charged sphere within a cylindrical pore; 106,390 nodes; 612,107 elements). Note: Others time including calculates relative residual and Jocobian of Newton method, initial CG and so on.

Mesh	Number of nodes	Number of elements
0	16,280	79,535
1	51,179	268,098
2	129,563	689,468
3	161,153	861,987
4	176,345	953,214

Table 3-4 The test mesh of number of nodes and elements for the second-order shape function.

	First-order	Second-order
Number of elements	634,603	79,535
Number of nodes	119,253	118,522 (init. 16,280)
Newton Iteration	5	5
CG Iteration	725	720
Total time (sec)	90.74	88.35

Table 3-5 Comparison of the first-order with the second-order shape function (CPU=6).

Level	Number of Nodes	Number of Elements	Force of Interaction (dimensionless)	Relative Inaccuracy (%)
0	924	3,821	20.9924	57.01%
1	3,333	15,810	34.6516	29.05%
2	19,171	101,644	40.4868	17.10%
3	32,103	169,825	45.1229	7.61%
4	36,149	192,478	47.0891	3.58%
5	36,316	193,368	48.4012	0.89%
6	-----	-----	-----	

Table 4-1 Evolution of adaptive mesh refinement for computing force of interaction between two free charged spheres (radius=5, λ=0.416, for *a posteriori* error estimator, CPU#6) use the first-order element.

Note: the force of interaction of finial mesh level of previously literature [Dyshlovenko, 2002] is 48.840.

Level	Number of nodes	Number of elements	Force of interaction (dimensionless)	Relative Inaccuracy (%)
0	924	3,821	28.3253	42.00%
1	3,245	14,867	40.1516	17.79%
2	18,366	98,253	45.3568	7.13%
3	24,626	135,443	48.0247	1.67%
4	27,492	158,463	48.5845	0.52%
5	28,012	159,359	48.8090	0.06%
6	28,095	159,403	48.8242	0.04%
7	-----	-----	-----	

Table 4-2 Evolution of adaptive mesh refinement for computing force of interaction between two free charged spheres (radius=5, λ=0.416, $\varepsilon_{pre} = 0.0003$ for *a posteriori* error estimator, CPU#6) use the second-order shape function.
Note: the force of interaction of finial mesh level of previously literature [Dyshlovenko, 2002] is 48.840.

Processor NO.	2	4	8	16	32	64
Preprocessing	12.5	13.5	14.4	15.4	19.5	18.2
I	92.1	24.8	6.2	2.0	1.5	2.4
II	977.9	267.4	68.5	23.2	10.2	9.9
III	0.9	1.8	2.0	3.3	3.8	6.9
IV	19.5	9.7	5.4	2.9	2.1	2.4
Total time	1103.0	317.2	96.5	46.7	37.1	39.8

I. Add nodes on cell edges
II. Renumber added nodes
III. Update connectivity data
IV. Build neighbor identifier array

Original mesh: 1,070,194 cells and 187,649 nodes.
Next refined mesh: 2,701,178 cells and 468,944 nodes.

Table 4-3 Timing (seconds) of PAMR module for different processor numbers.

Level	Number of nodes	Number of elements	Force (Newtons)
0	16,280	79,535	5.432e-16
1	51,179	268,098	3.422e-16
2	129,563	689,468	1.343e-16
3	161,153	861,987	9.433e-15
4	176,345	953,214	8.942e-15
5	185,688	1,014,692	8.411e-15
6	185,697	1,014,730	8.325e-15

Table 4-4 Evolution of force interaction between two identical charged spheres near a like-charged flat plate (h=2.5μm, r=3μm, a=0.325μm and spheres Ψ_0=3.0, flat plate Ψ_0=5.0).

Level	Number of nodes	Number of elements	Force (Newtons)
0	17,058	83,588	-8.433e-15
1	52,549	275,534	-4.645e-15
2	131,067	698,093	-1.343e-15
3	161,345	872,534	-9.842e-16
4	177,269	963,419	-7.739e-16
5	184,564	1,019,692	-6.178e-16
6	184,611	1,020,243	-5.491e-16

Table 4-5 Force of interaction with different refinement level mesh in the simulation of two identical charged spheres near a charged flat plate (h=2.5, r=4.5, a=0.325) for the first-order element.

Level	Number of nodes	Number of elements	Force (Newtons)
0	17,058	83,588	-4.681e-15
1	39,150	200,756	-9.563e-16
2	94,365	485,608	-7.972e-16
3	101,803	539,542	-7.319e-16
4	102,433	541,478	-6.931e-16
5	102,445	541,503	-6.817e-16
6	----	----	----

Table 4-6 Force of interaction with different refinement level mesh in the simulation of two identical charged spheres near a charged flat plate (h=2.5, r=4.5, a=0.325) for the second-order shape function.

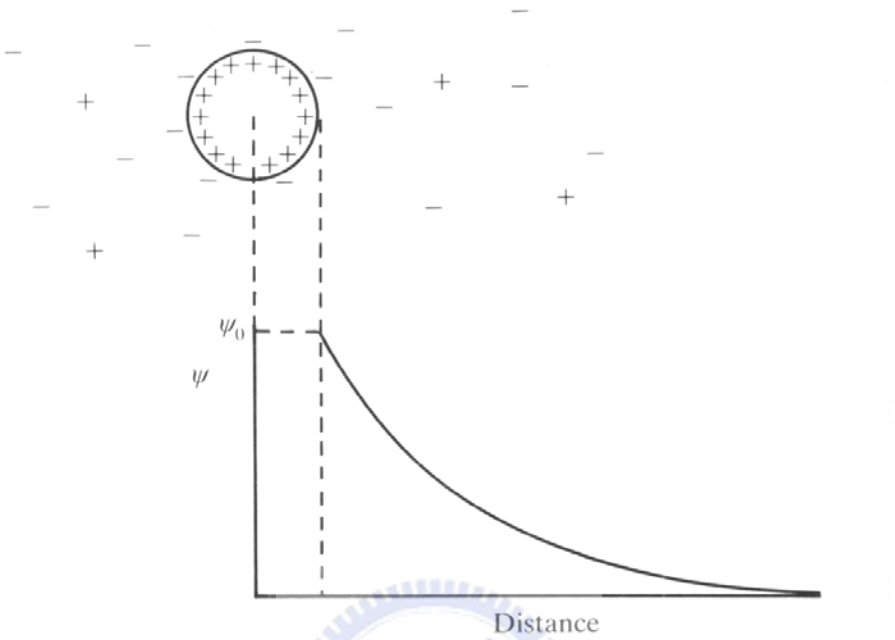


Fig. 1-1 Electrostatic potential near a positively charged colloidal particle.

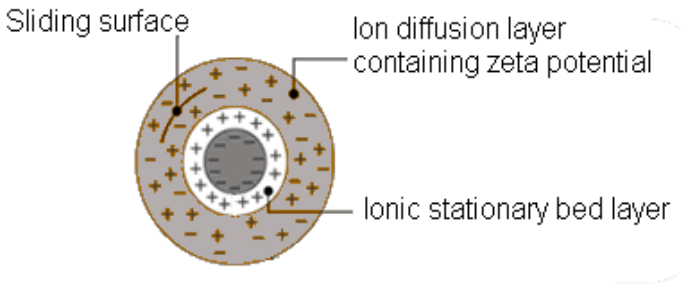


Fig. 1-2 The illustration of colloidal particle.

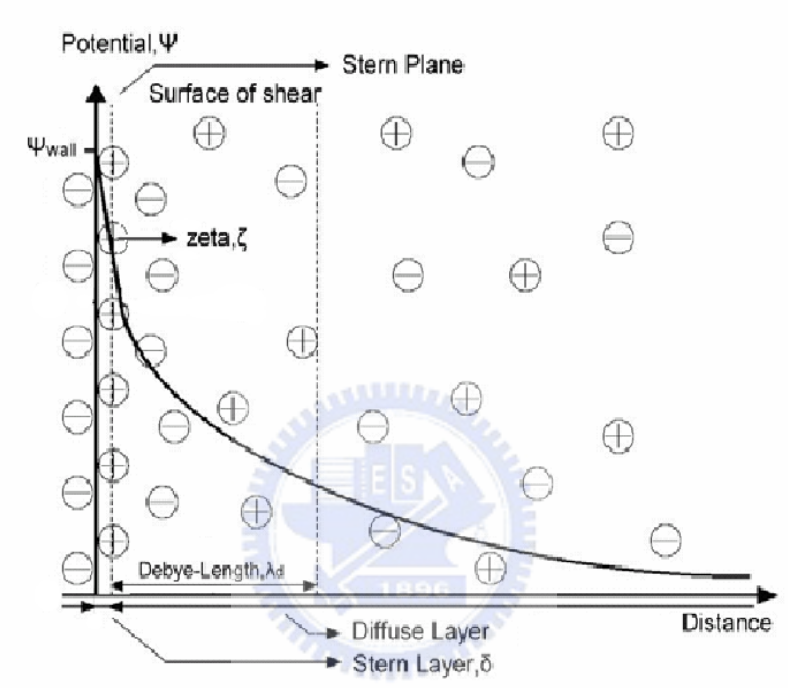


Fig. 1-3 The Electric Double Layer distribution.

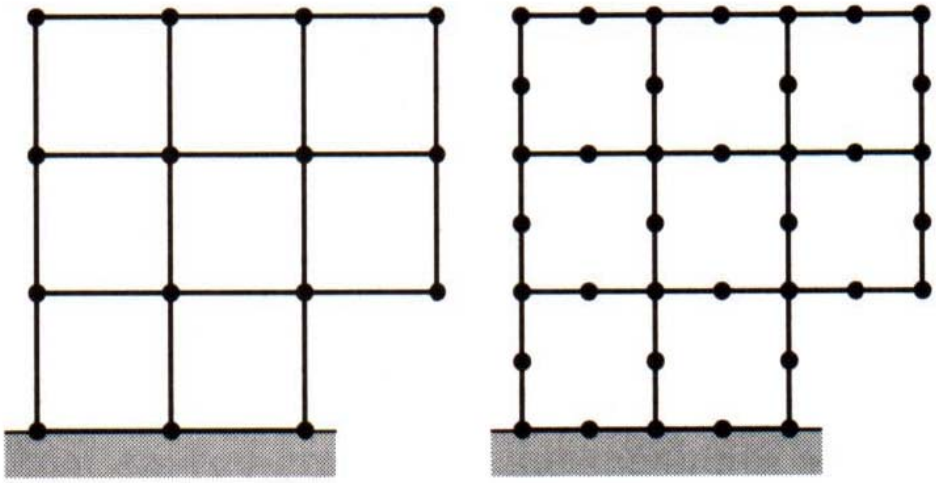


Fig. 1-4 *p*-refinement.

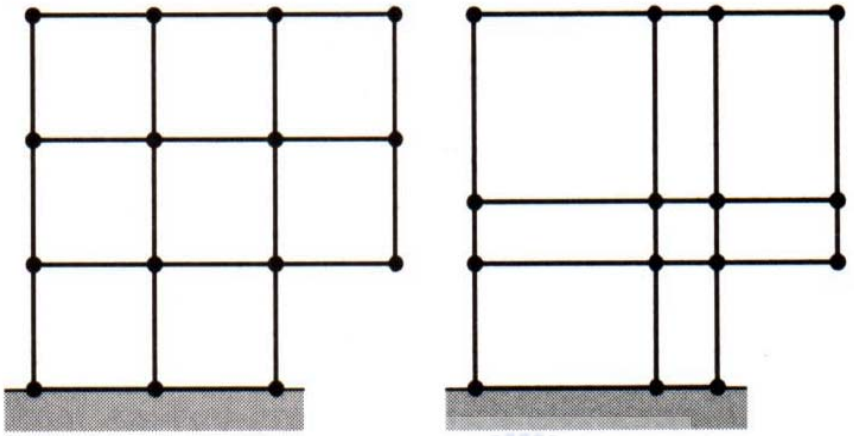


Fig. 1-5 *r*-refinement.

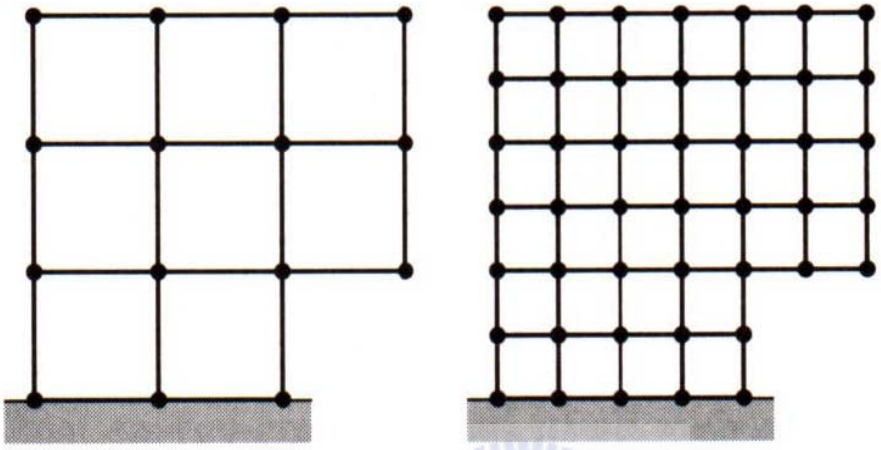


Fig. 1-6 *h*-refinement.

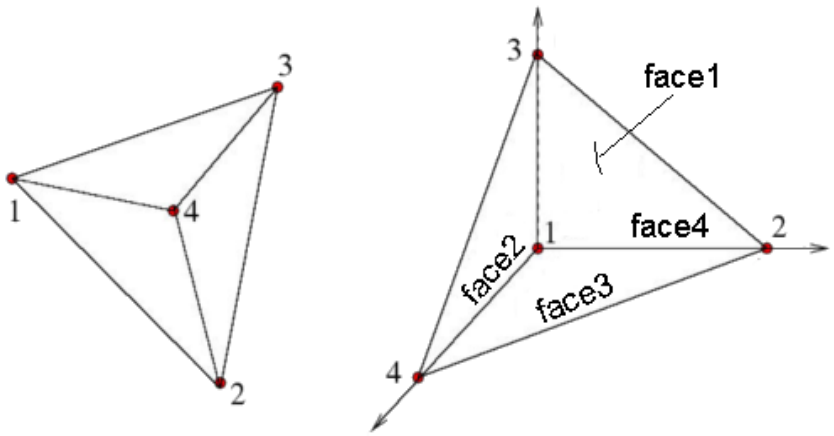


Fig. 2-1 Three-dimensional tetrahedral element and face.

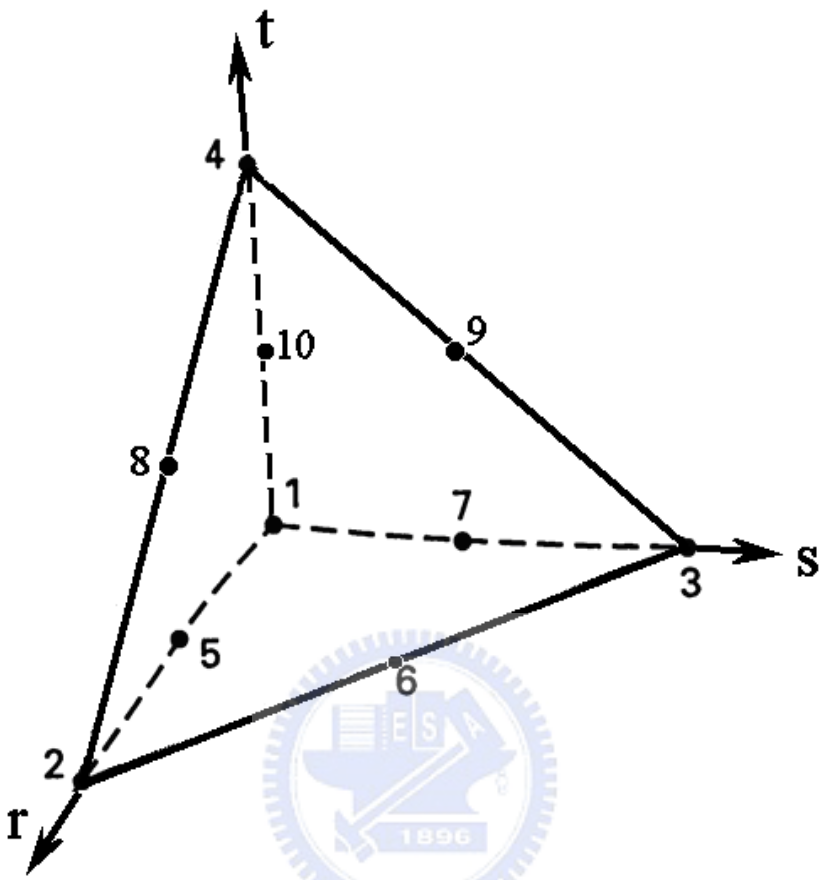


Fig. 2-2 Interpolation functions of ten variable-number-nodes three-dimensional tetrahedral element.

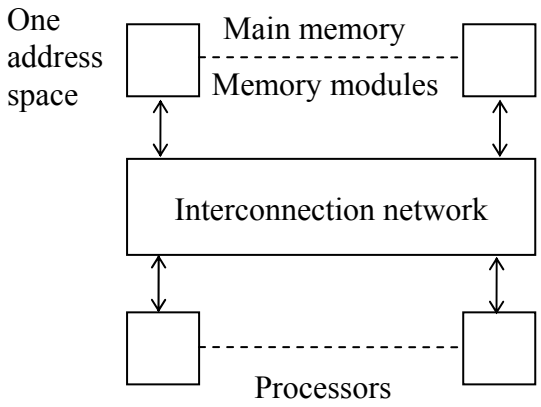


Fig. 2-3 Traditional shared memory multiprocessor model.

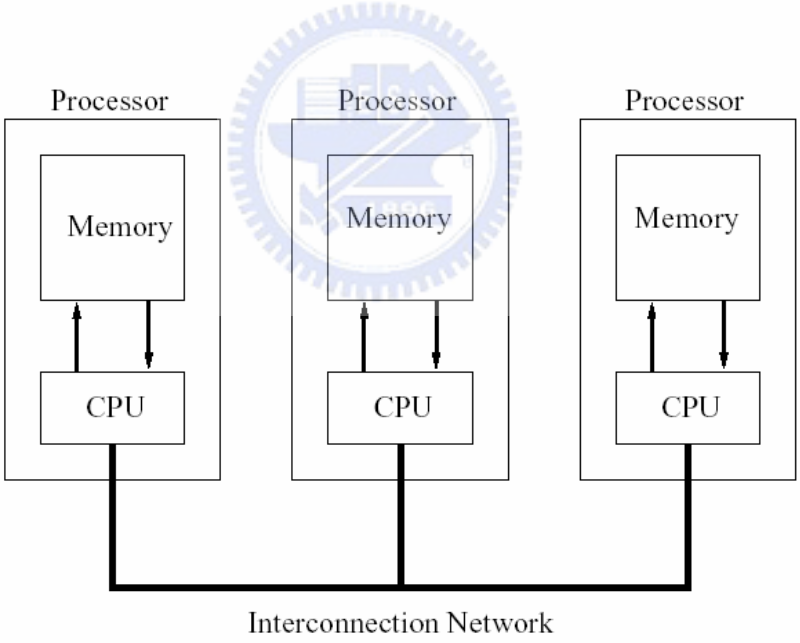


Fig. 2-4 Distributed memory multiprocessor architecture.

Fig. 2-5 PC cluster at MuST.

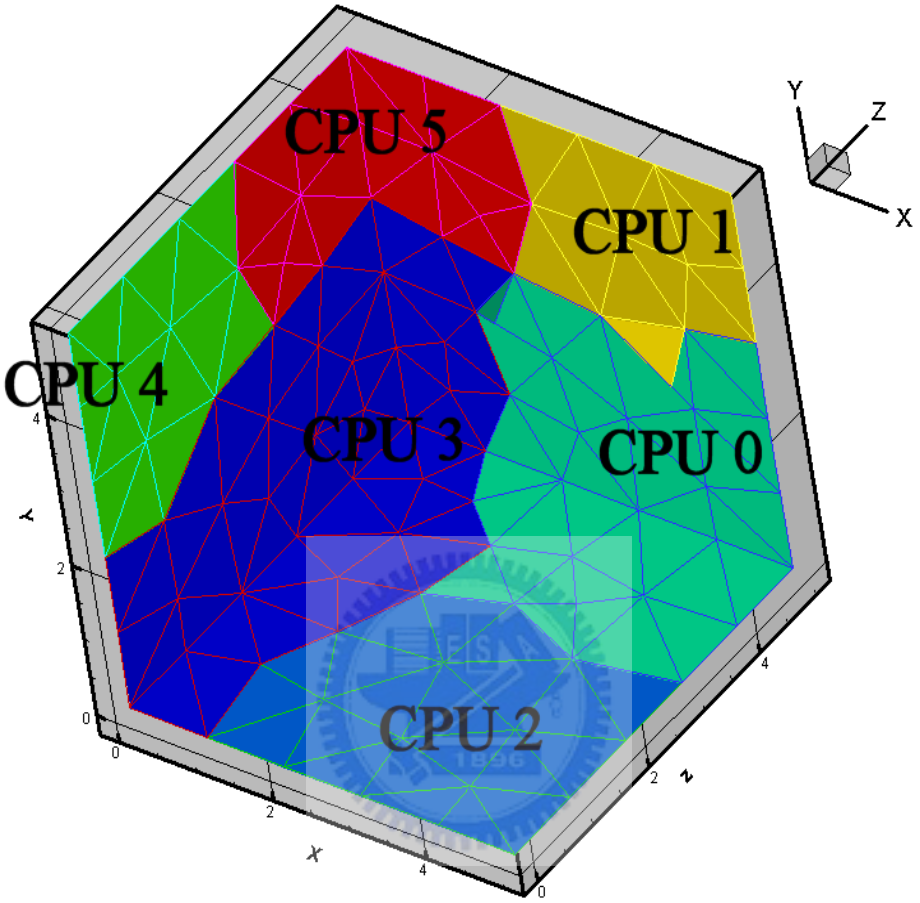


Fig. 2-6 The Poisson-Boltzmann equation solver by subdomains.

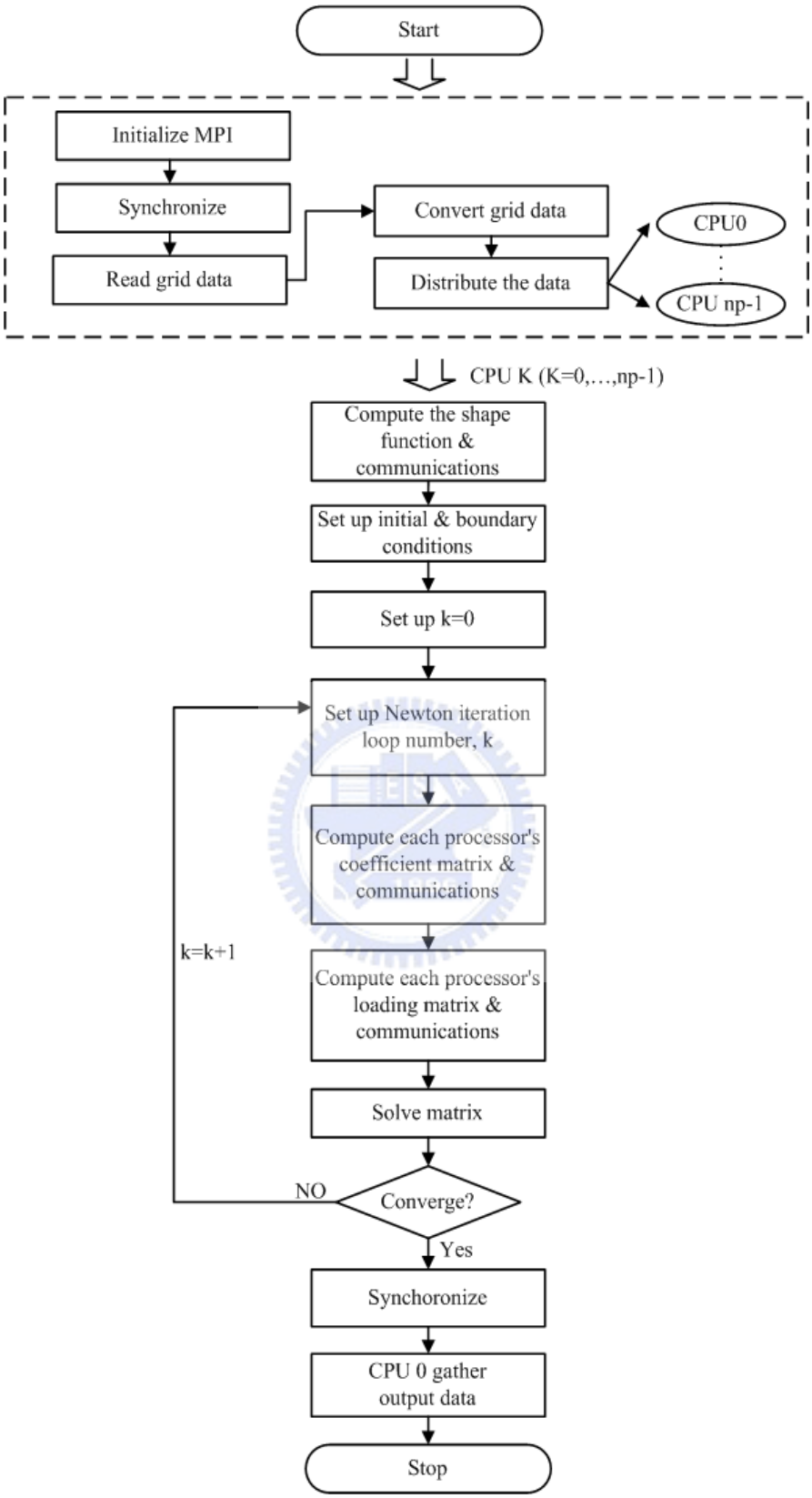


Fig. 2-7 The flow chart of parallel Poisson-Boltzmann equation solver.

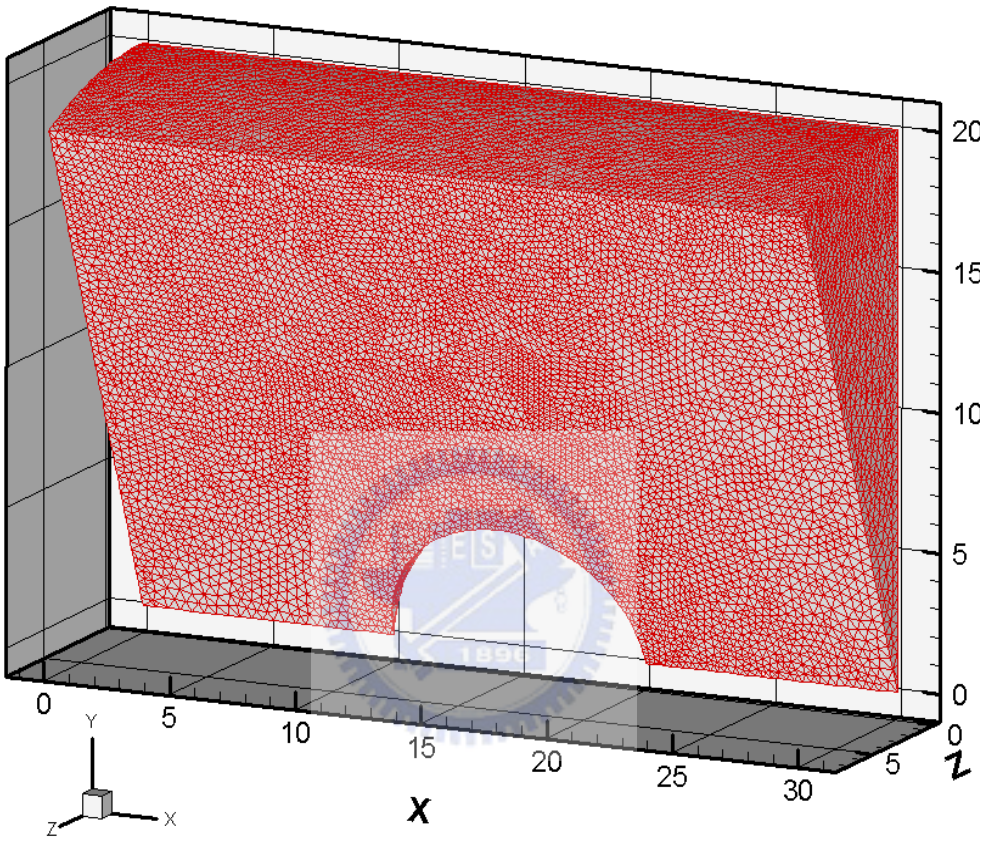
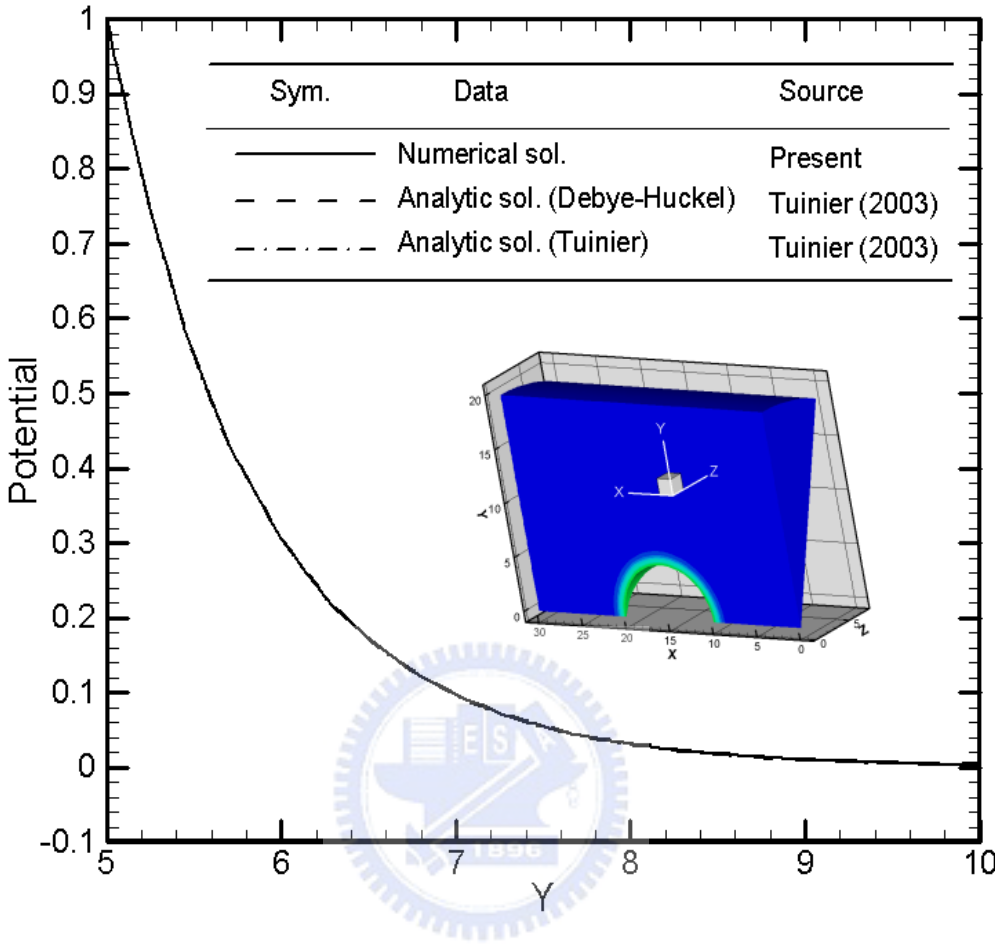
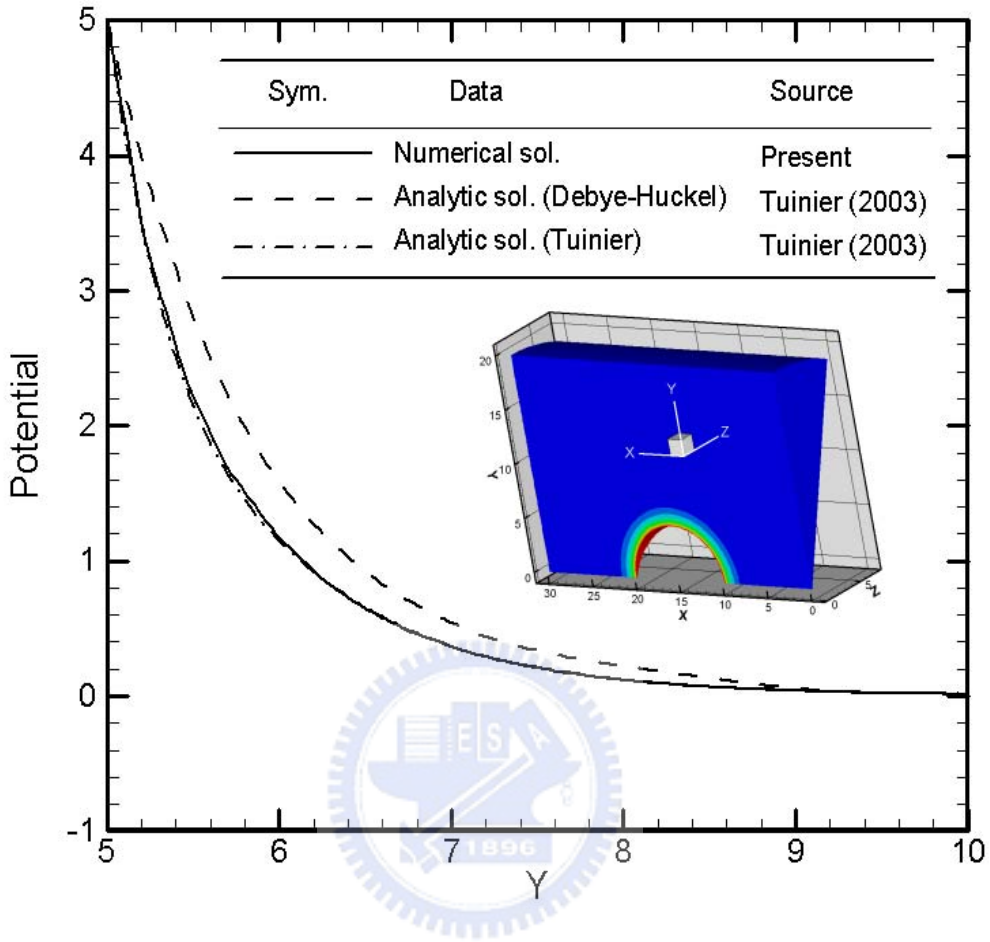


Fig. 3-1 The surface mesh distribution of the sphere (176,309 elements; 36,396 nodes).


Sym.
Data
Source
Numerical sol.
Present
Analytic sol. (Debye-Huckel)
Tuinier (2003)
Analytic sol. (Tuinier)
Tuinier (2003)
Potential
Y
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
5
6
7
8
9
10

(a)



(b)

Fig. 3-2 Comparison of potential distributions around a charged sphere (a) surface potential Ψ_0=1.0 (b) surface potential Ψ_0=5.0 (18,253 nodes; 96,638 elements).

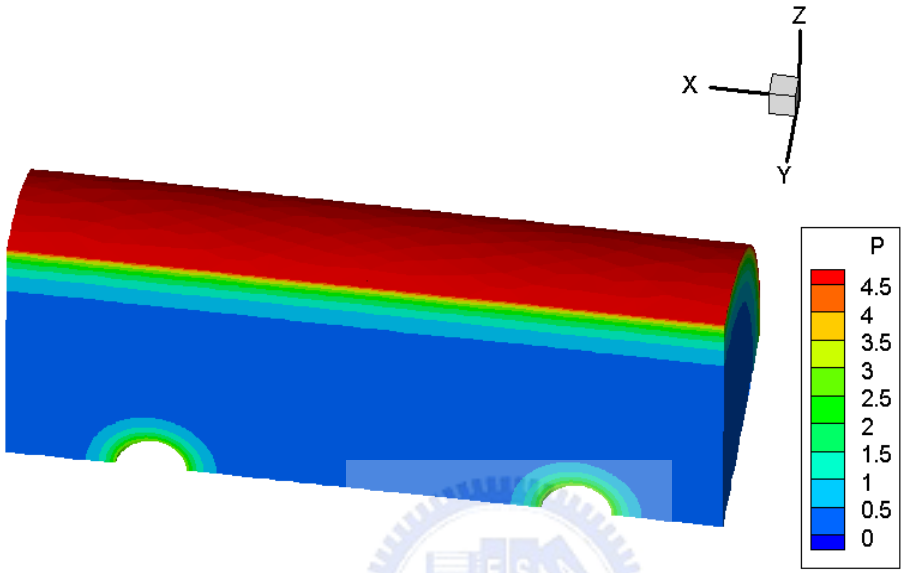


Fig. 3-3 The potential distribution for the sphere confined in a pore at separation distance r=6.

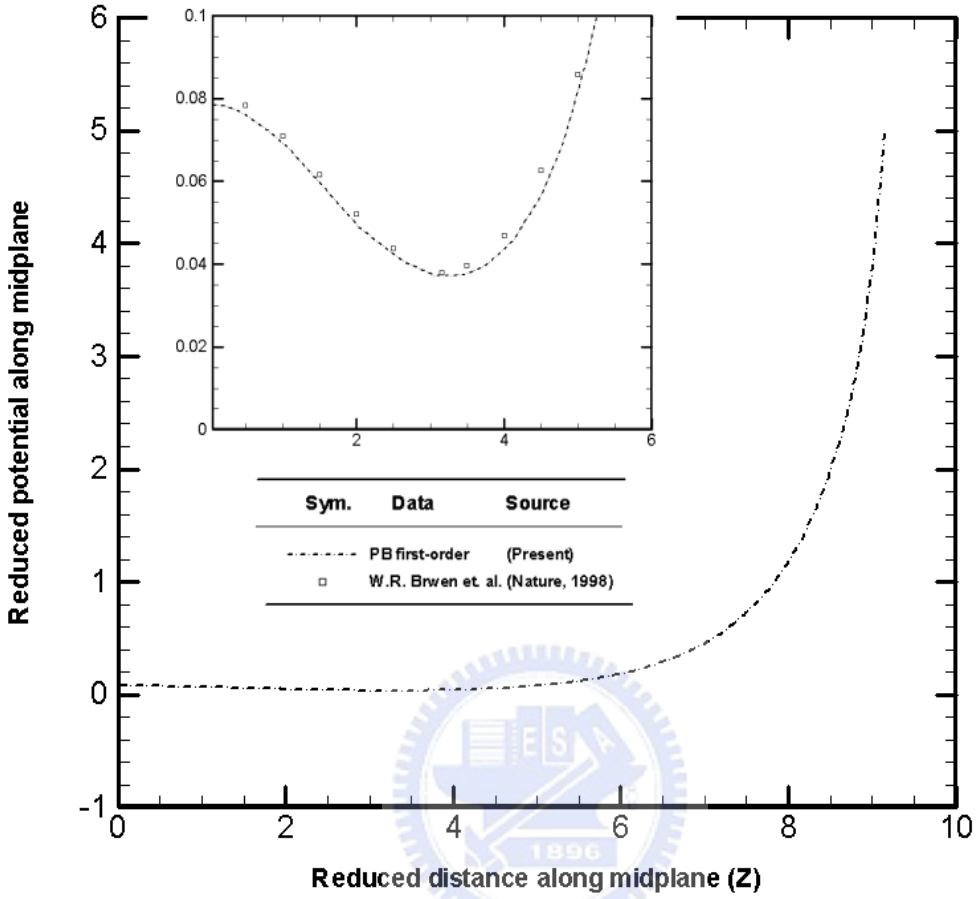


Fig. 3-4 Potential along midplane between two spheres (r=6).

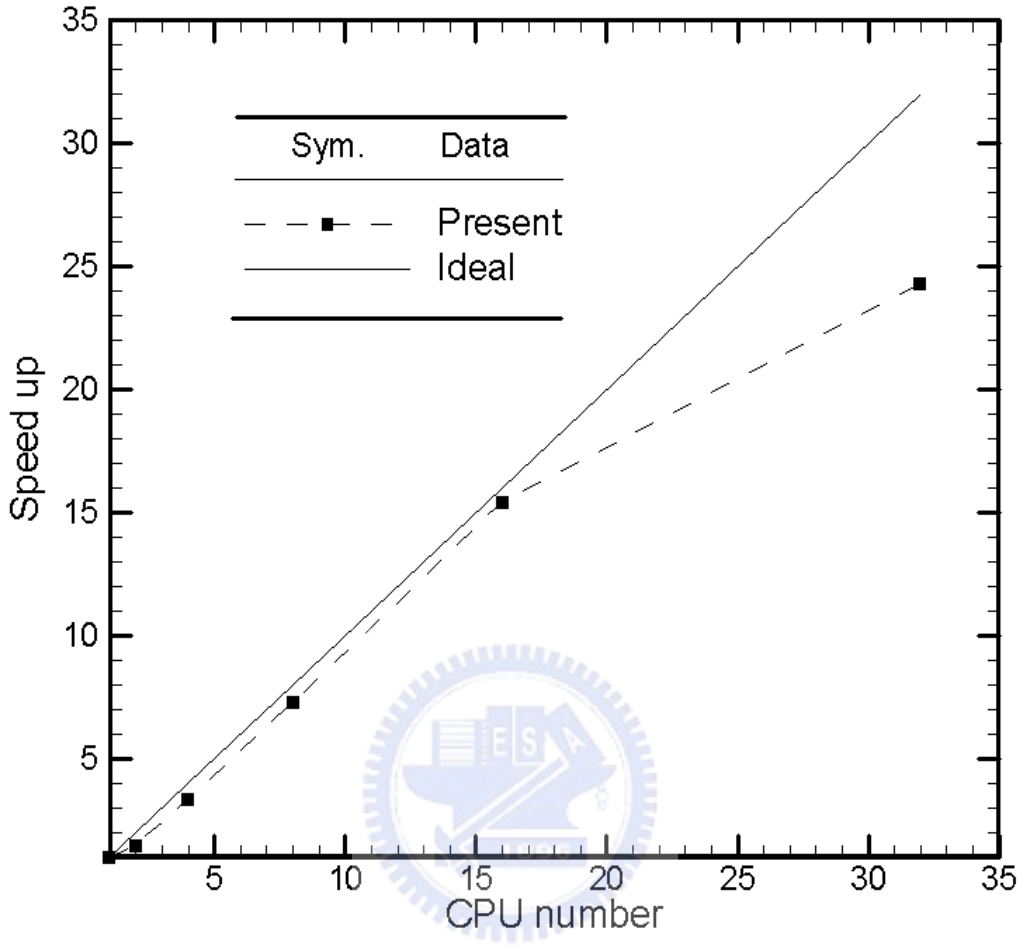
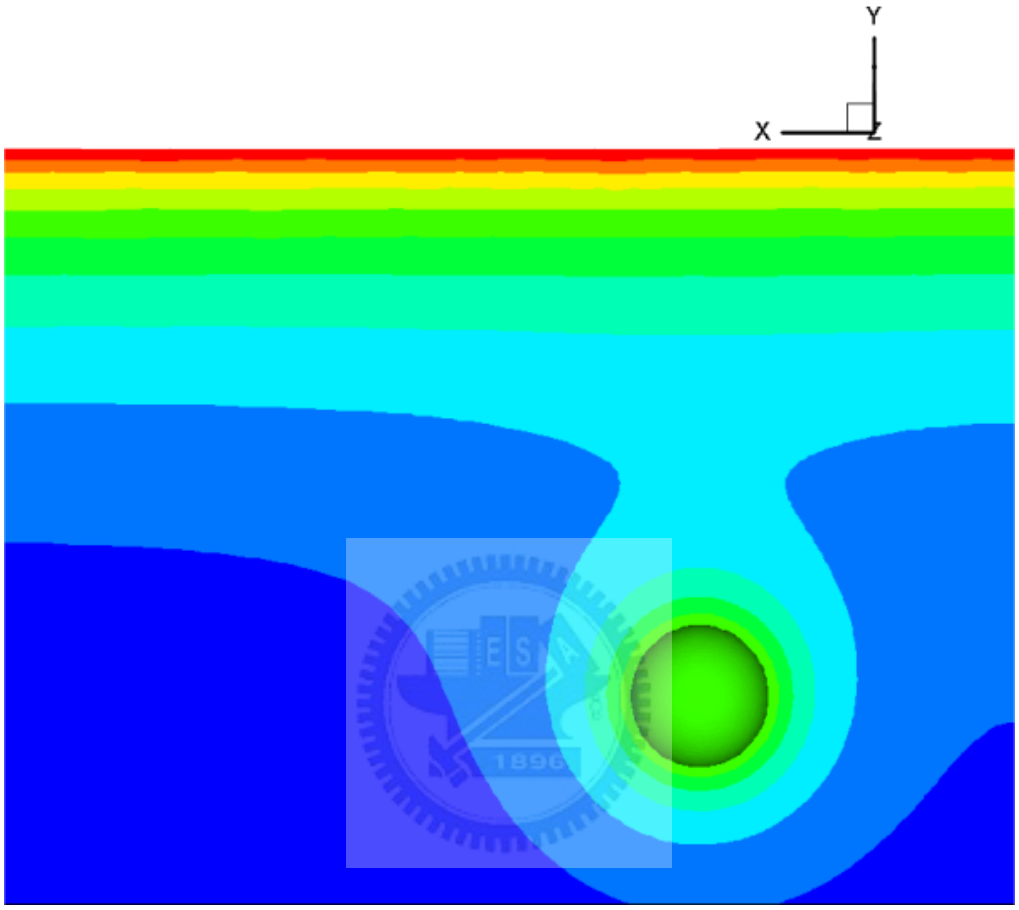
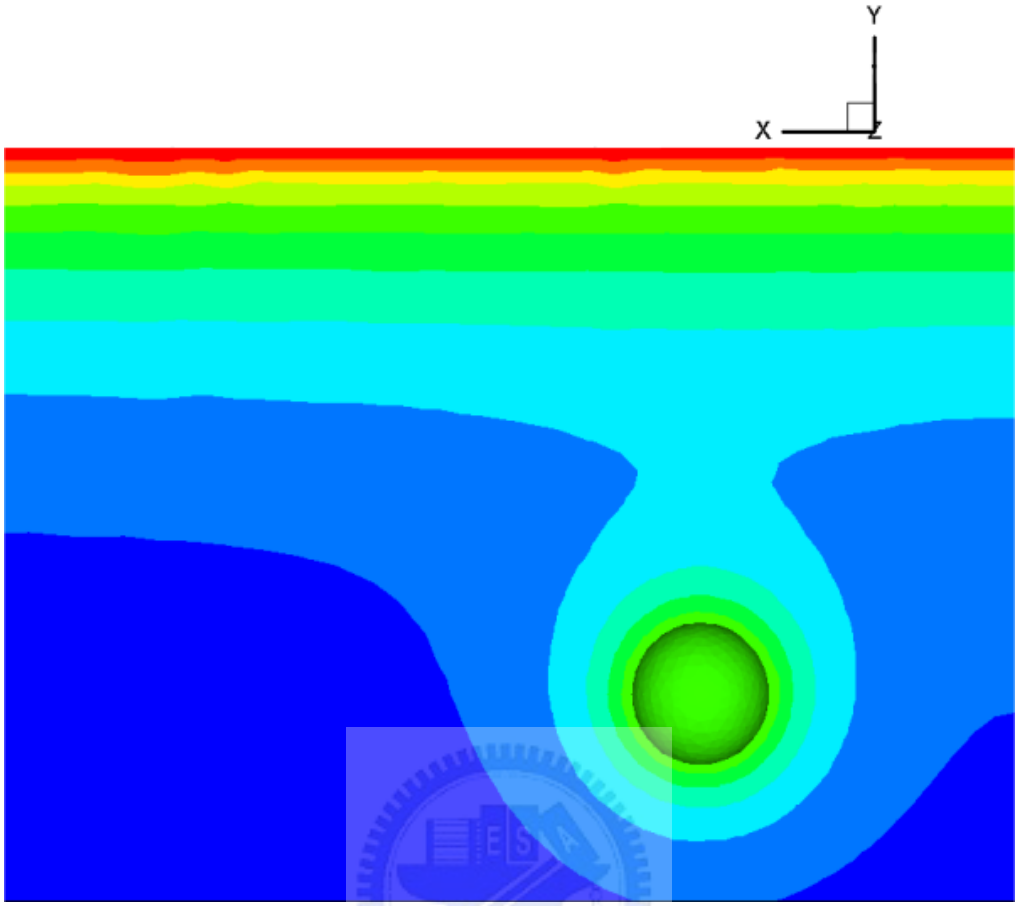


Fig. 3-5 The parallel speedup for the parallelized Poisson-Boltzmann equation solver for simulating the potential distribution around a charged sphere (radius=5, Ψ_0 =2.0) within a cylindrical pore (106,390 nodes; 612,107 elements).



(a)



(b)

Fig. 3-6 The distribution of potential around the charged spheres (radius a=0.325μm) near a like-charged plate (h = 2.5μm) at level-0 for solutions of (a) the first-order element (b) the second-order shape function.

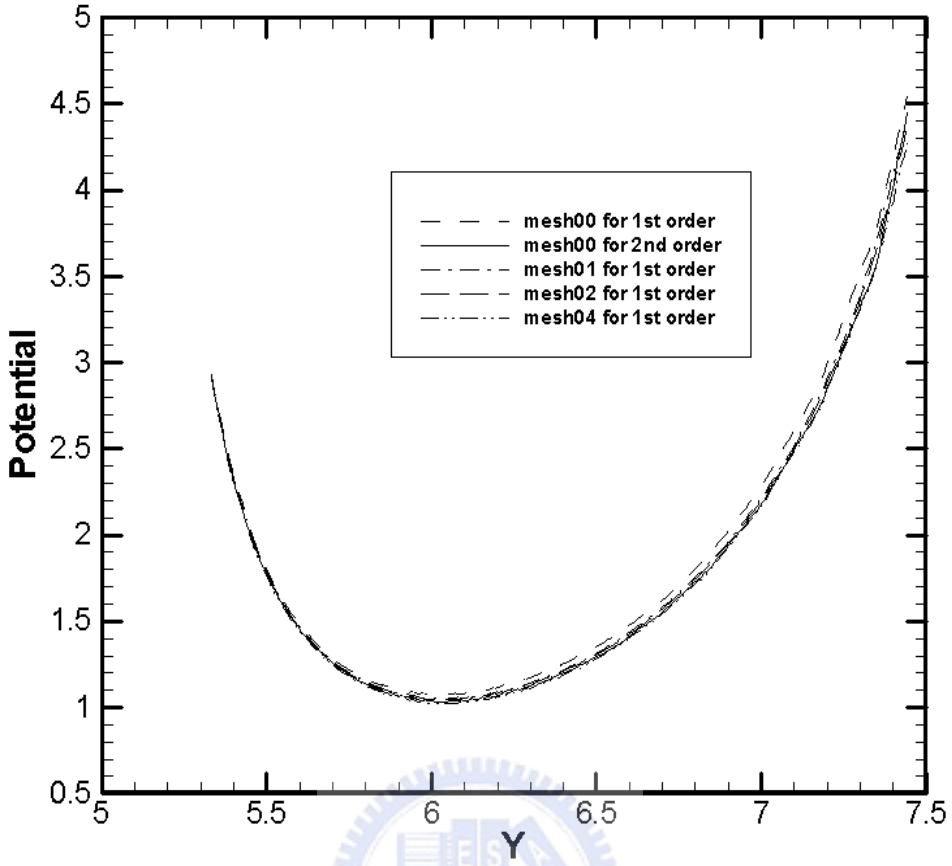


Fig. 3-7 The profiles of different mesh levels of the first-order element and the second-order shape function. the distribution of potential around the charged spheres (radius a=0.325μm) near a like-charged plate (distance h = 2.5μm).

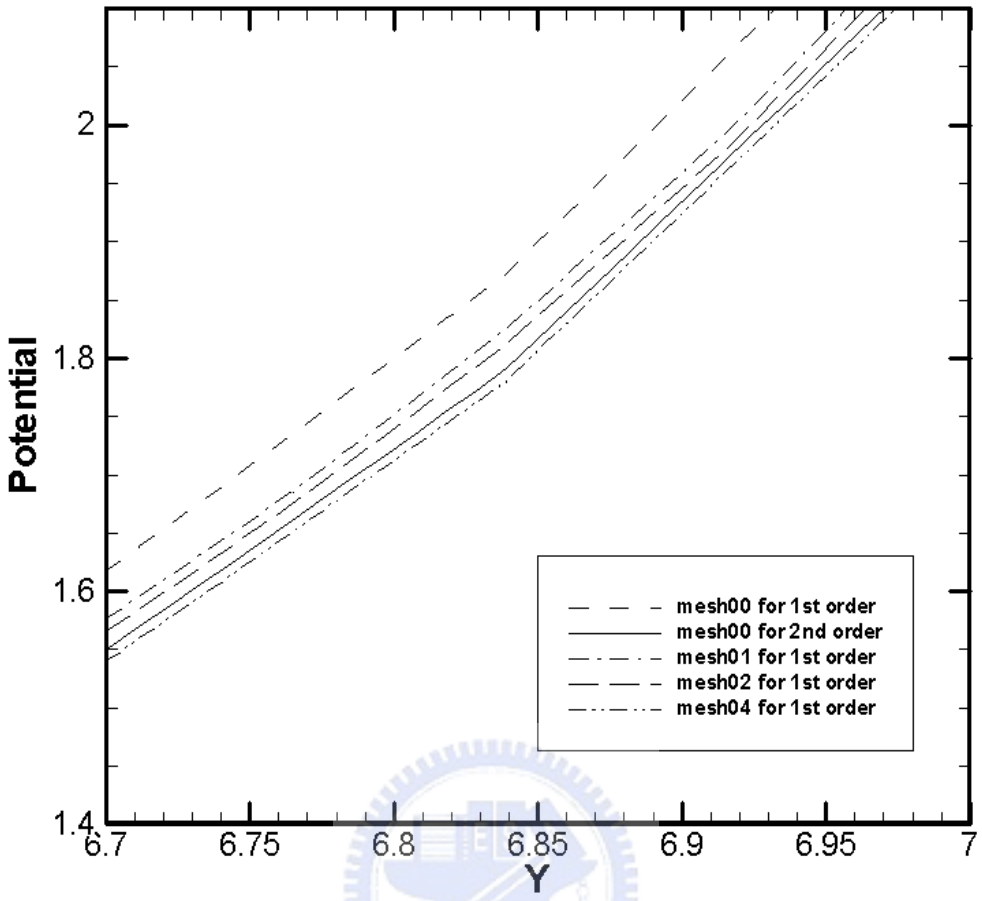


Fig. 3-8 The profiles of different mesh levels of the first-order element and the second-order shape function (in local). the distribution of potential around the charged spheres (radius a=0.325μm) near a like-charged plate (distance h = 2.5μm).

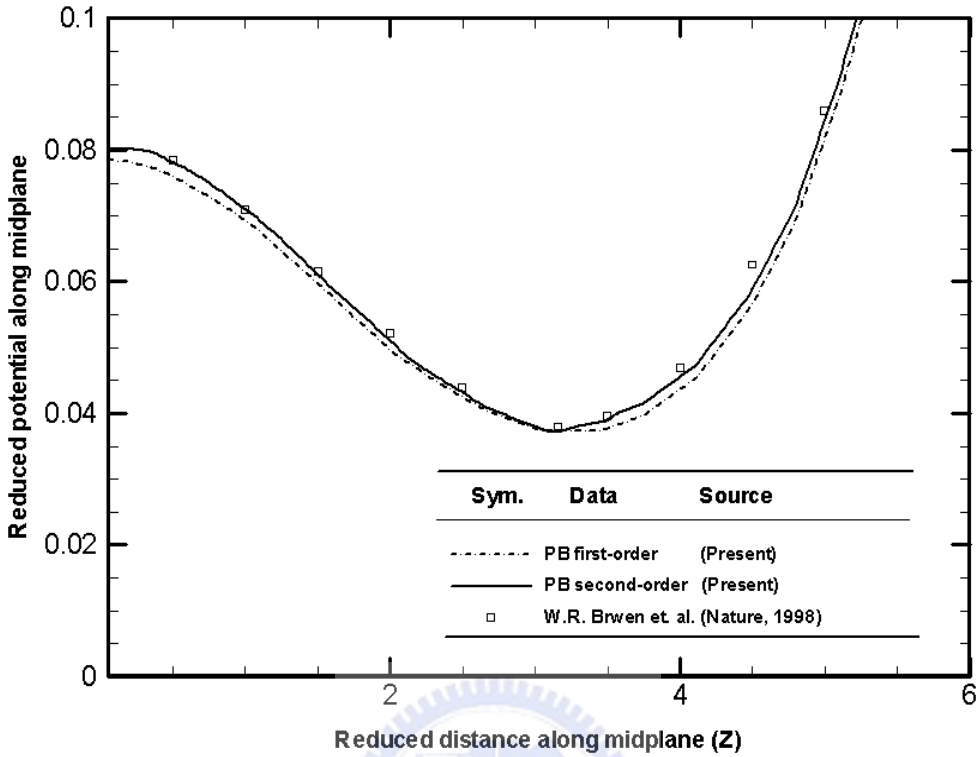


Fig. 3-9 Potential along midplane between two spheres (r=6), compare the first-order element with the second-order shape function. (nodes 42,308 and elements 246,155).

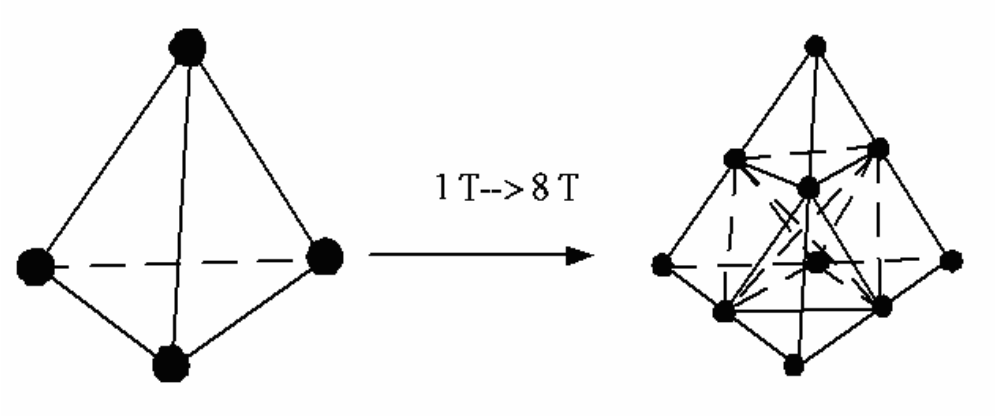


Fig. 4-1 Isotropic mesh refinement of tetrahedral mesh (T: Tetrahedron).

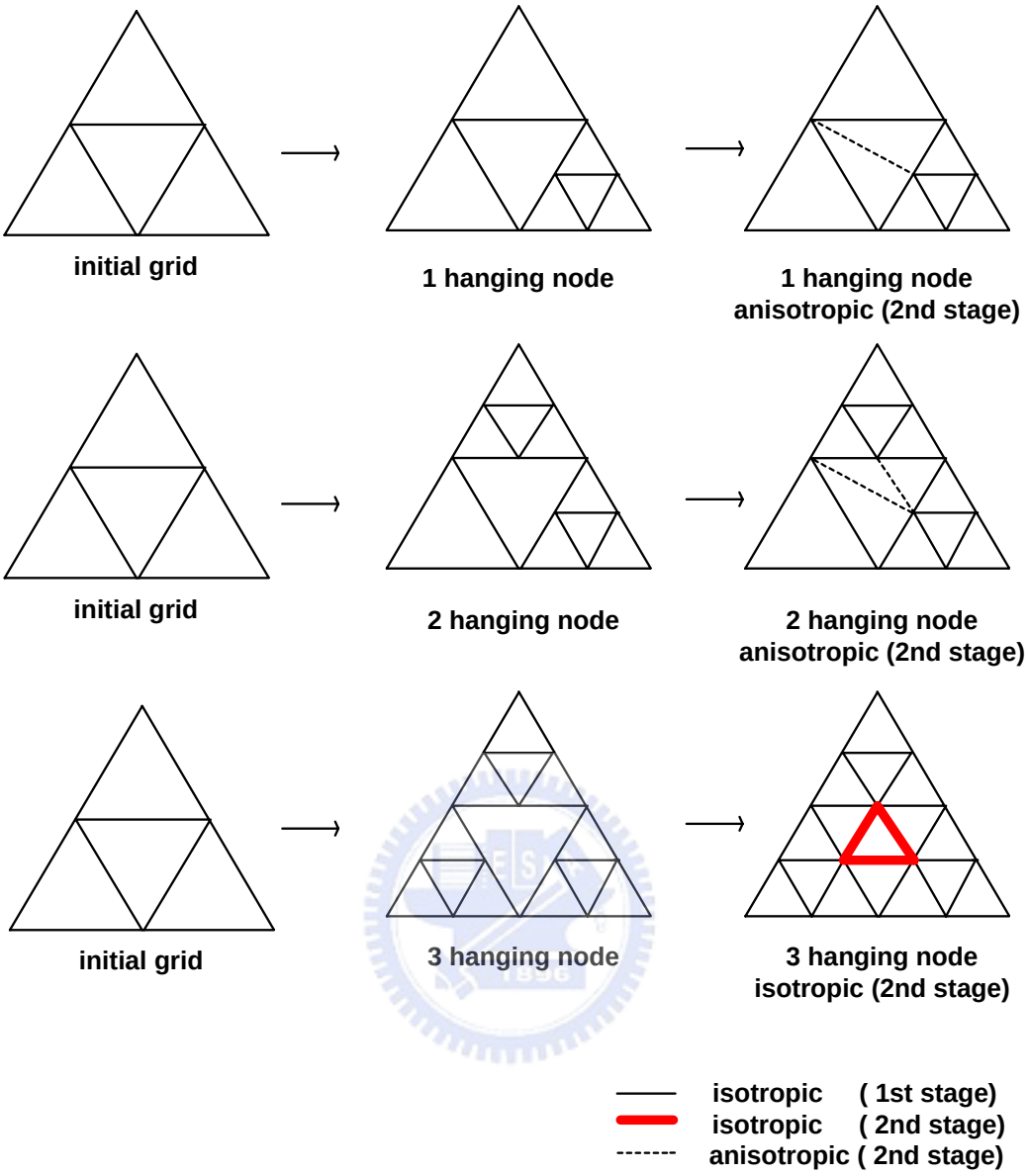


Fig. 4-2 Mesh refinement rules for two-dimensional triangular cell.

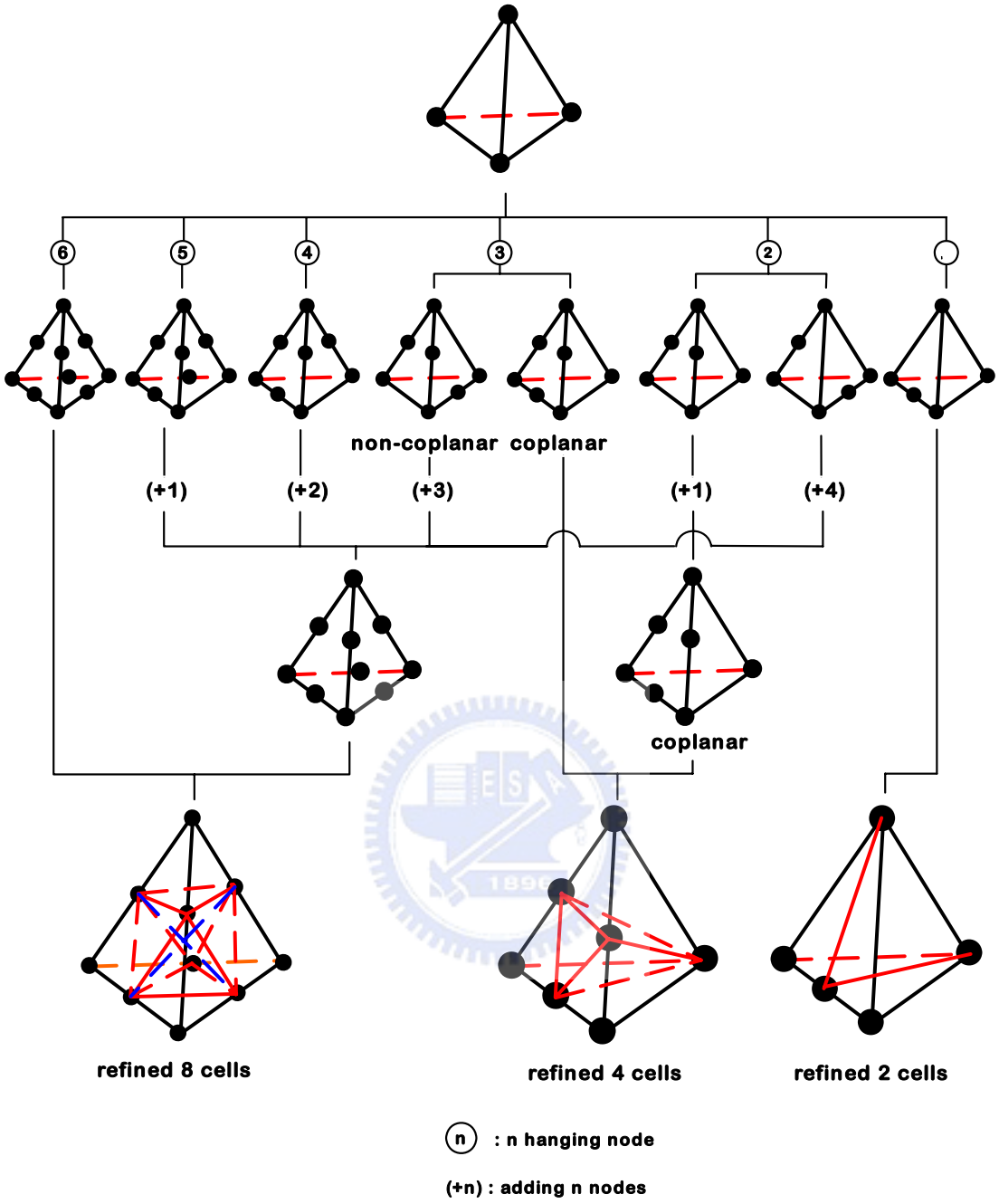
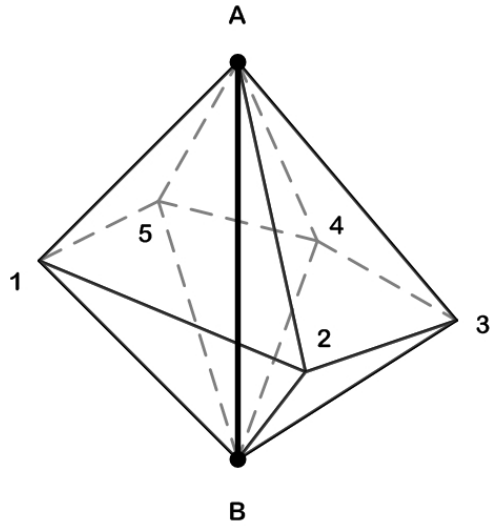


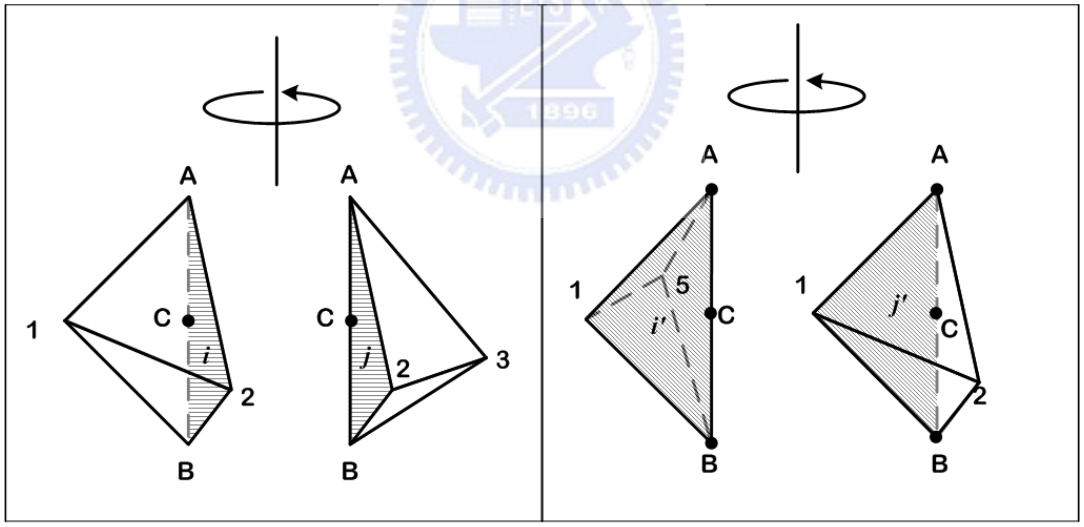
Fig. 4-3 Schematic diagram for mesh refinement rules of tetrahedron.



Common Edge A-B is shared by the following five cells:

A-B-1-2, A-B-2-3, A-B-3-4, A-B-4-5, A-B-5-1

(a)



(b) (c)

Fig. 4-4 Example of the common edge shared by the neighboring cells.

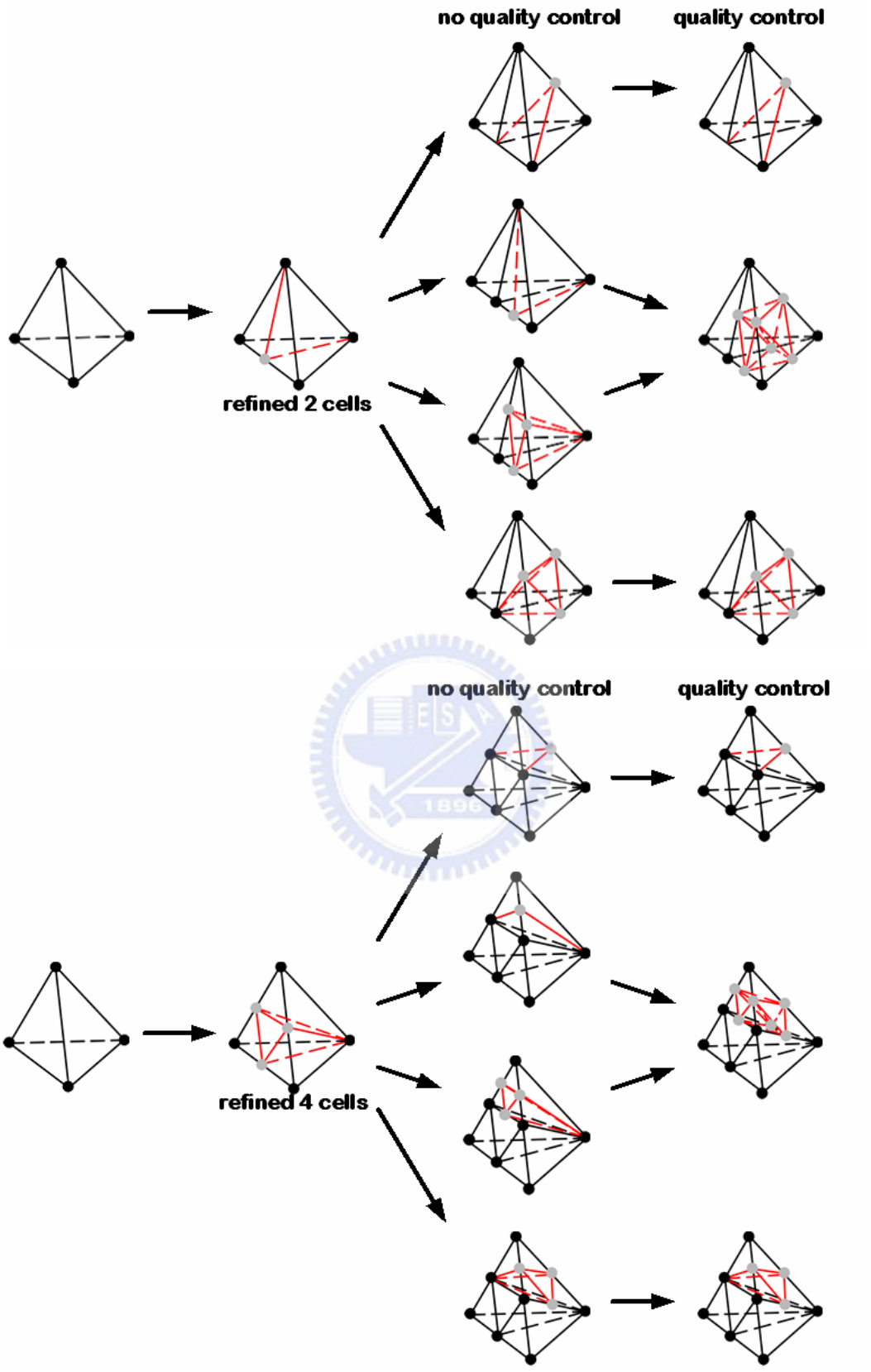


Fig. 4-5 Schematic diagram of the proposed cell quality control.

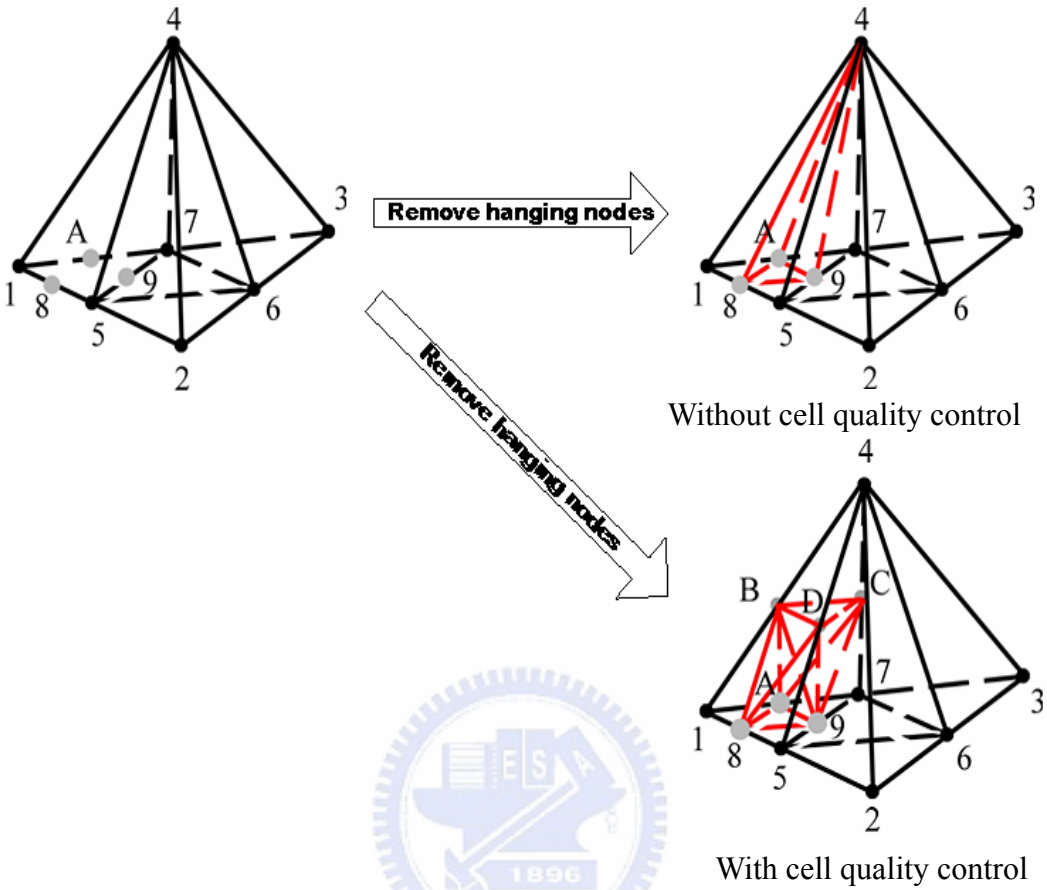


Fig. 4-6 Schematic diagram of typical cell quality control.

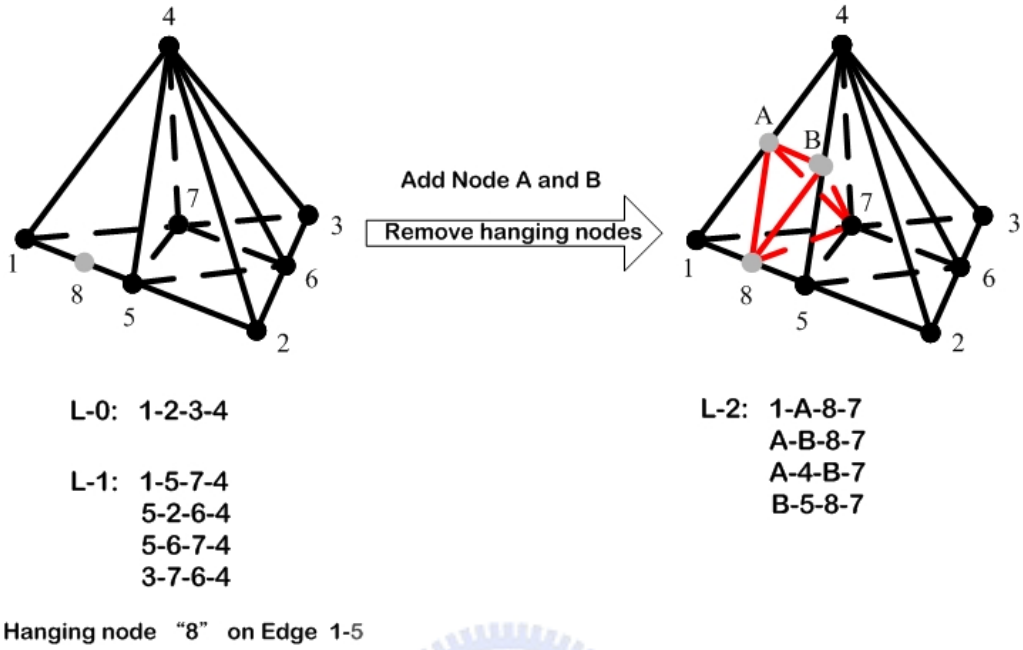


Fig. 4-7 A case that the proposed cell-quality-control would not affect to it.

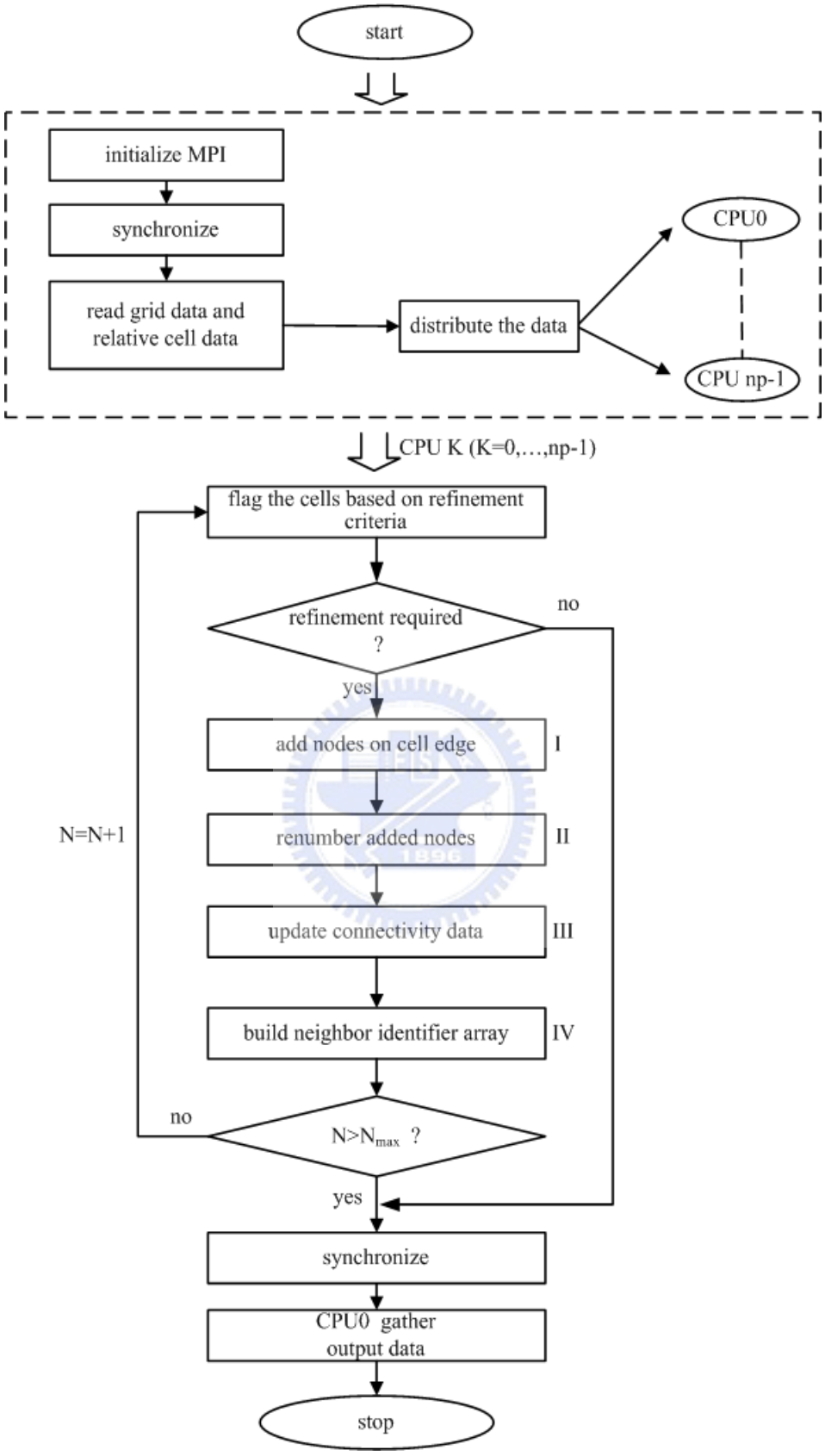


Fig. 4-8 Flow chart of parallel mesh refinement module.

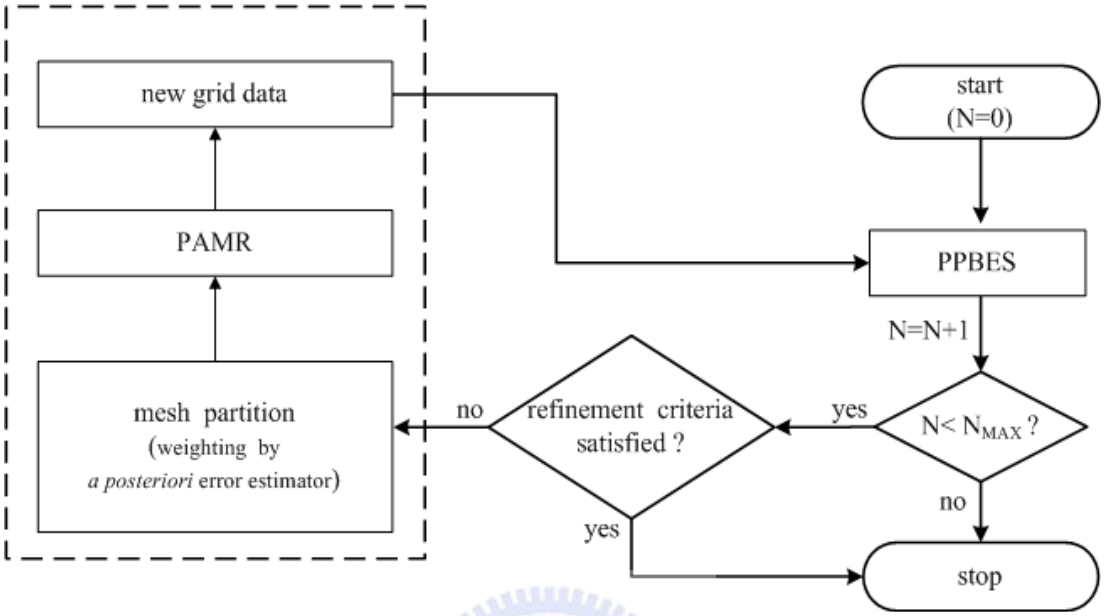
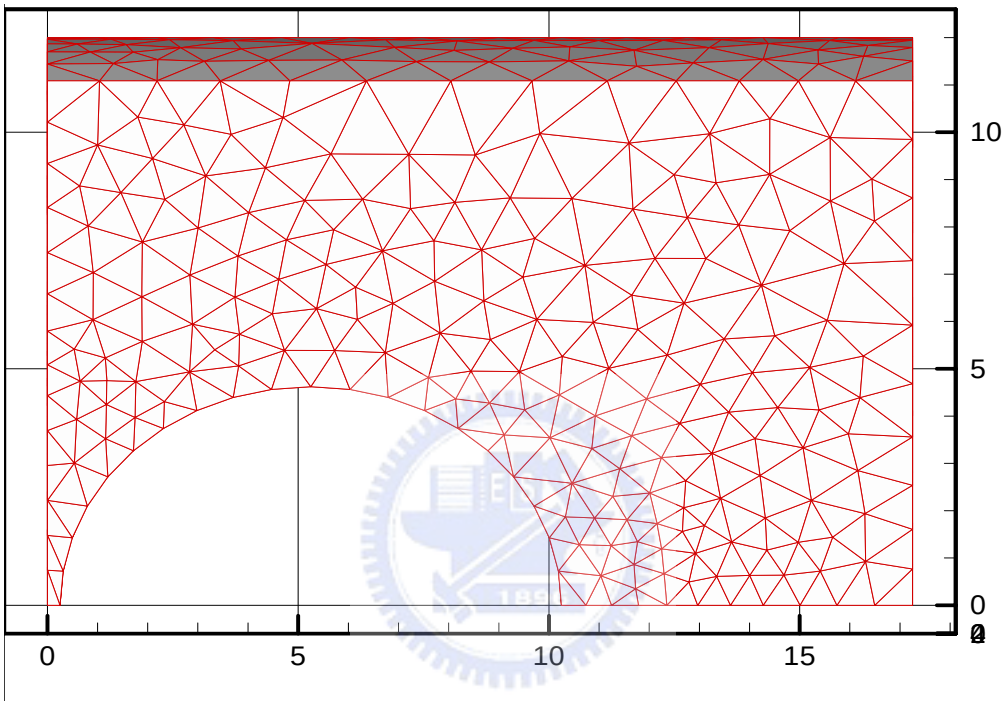
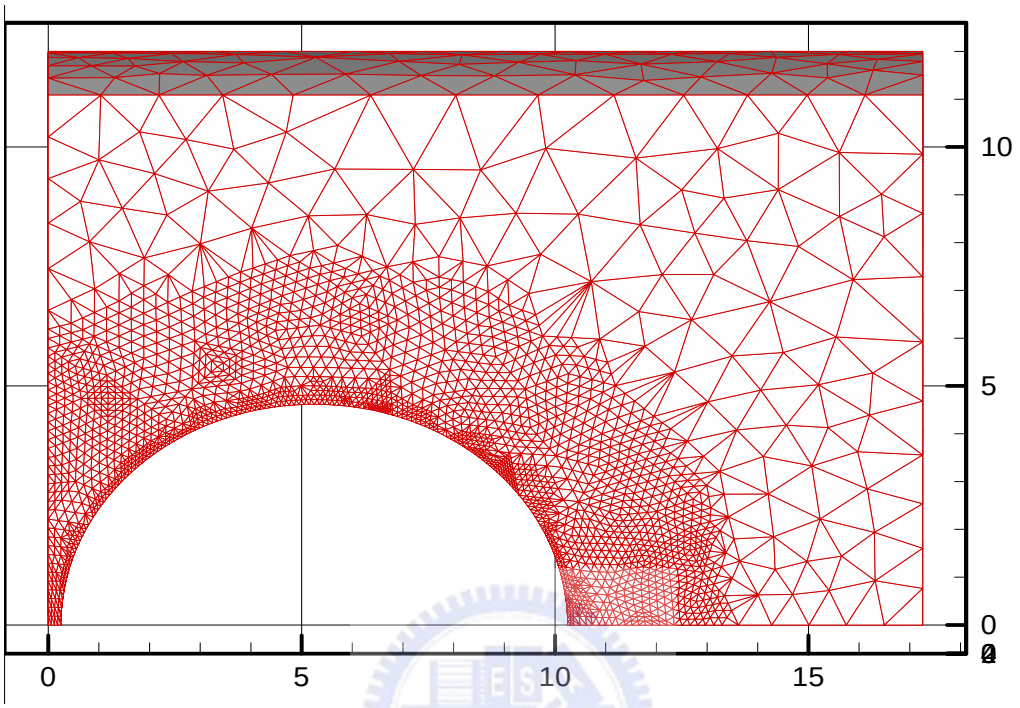


Fig. 4-9 Coupled PPBS-PAMR method.

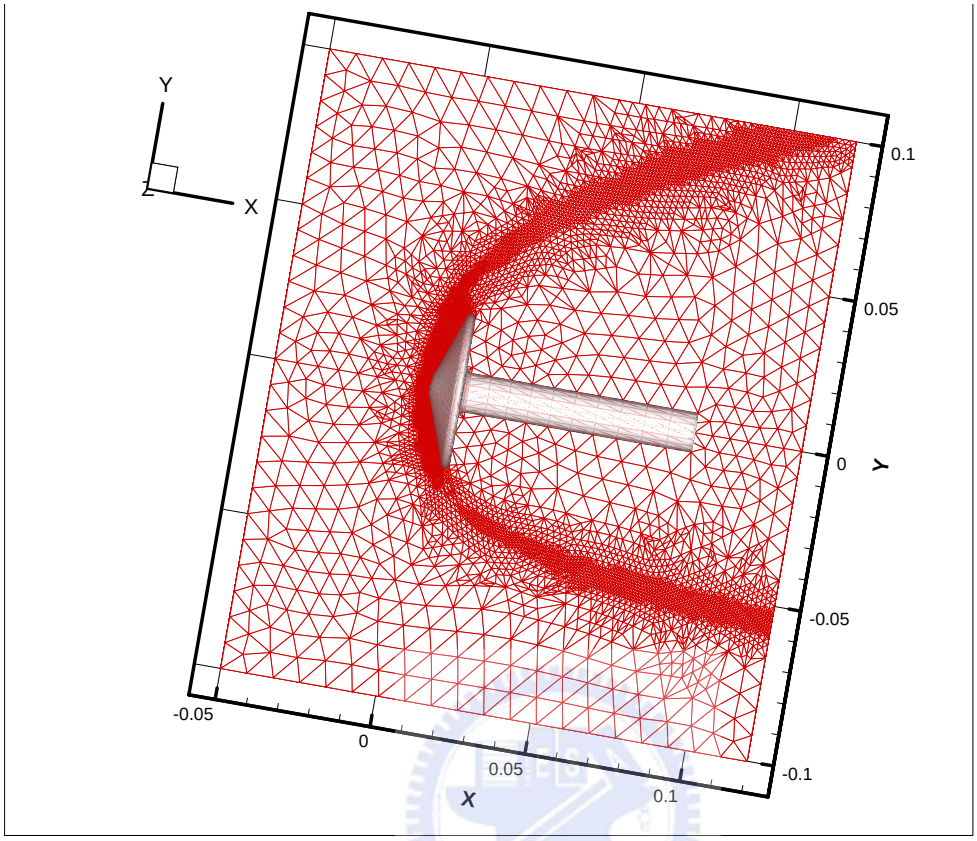


(a)

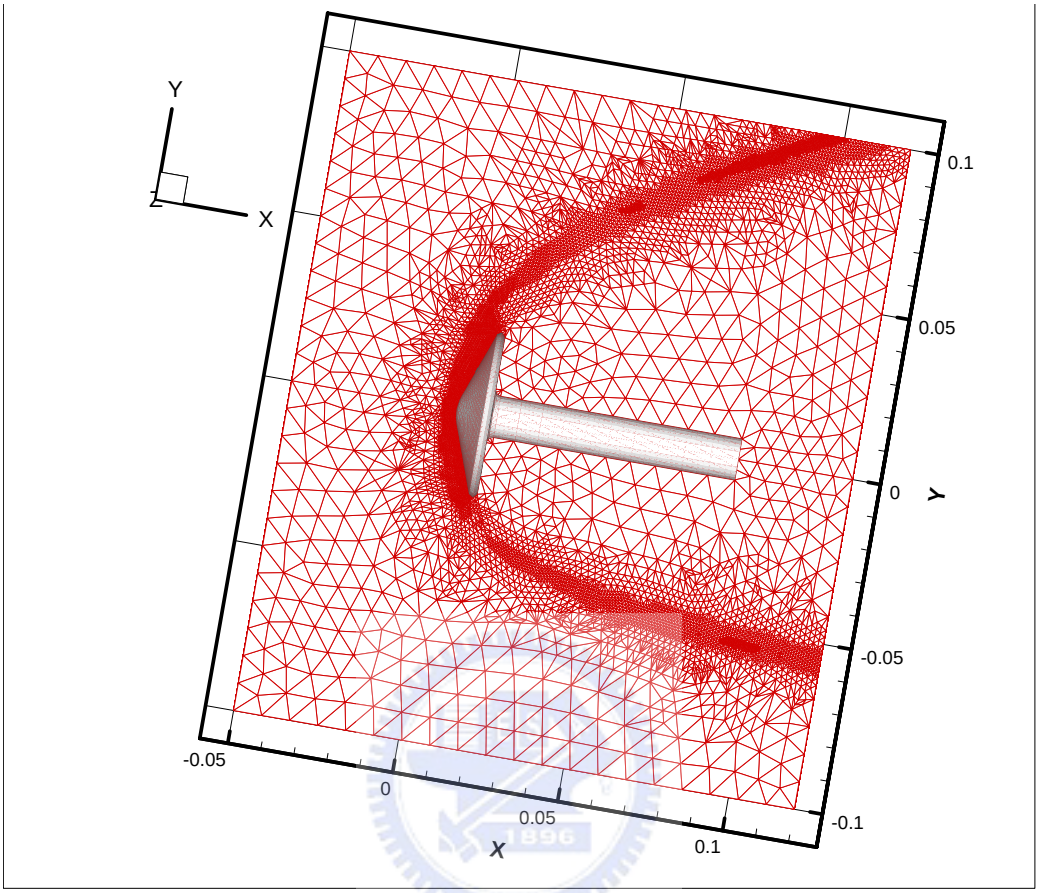


(b)

Fig. 4-10 The surface mesh distribution between two free identical charged spheres (radius=5, λ=0.416) at (a) level-0 (initial; 924 nodes; 3,821 elements) (b) level-5 (final; 36,316 nodes; 193,368 elements) mesh refinement.



(a)



(b)

Fig. 4-11 Original (Level-3, 1,070,194 tetrahedral cells and 187,649 nodes) mesh distribution and next (Level-4, 2,701,178 cells and 468,944 nodes) refined mesh.

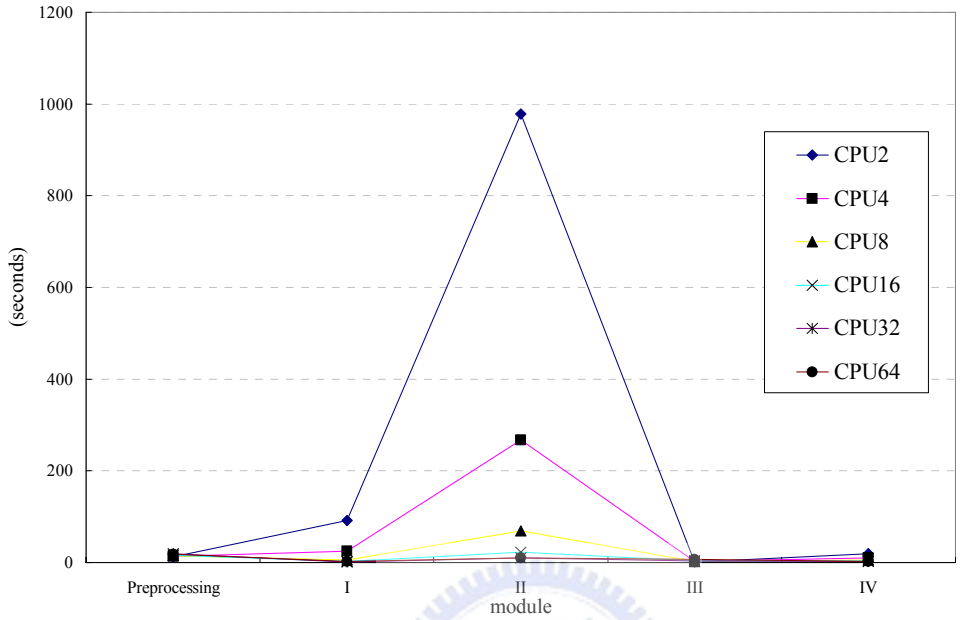


Fig. 4-12 Timing for different modules of PAMR at different processors

Note that module.

I. Add nodes on cell edges
II. Renumber added nodes
III. Update connectivity data
IV. Build neighbor identifier array

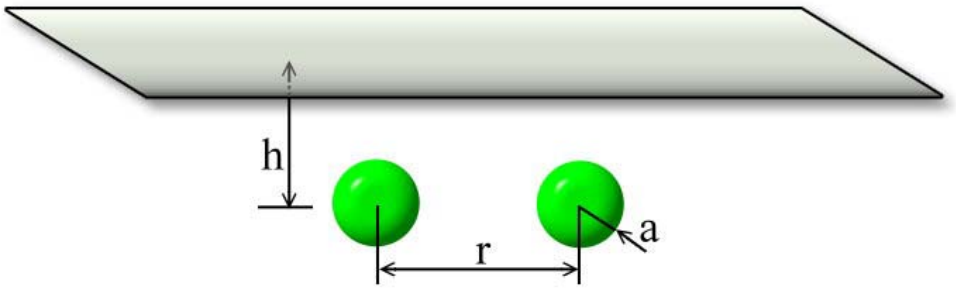
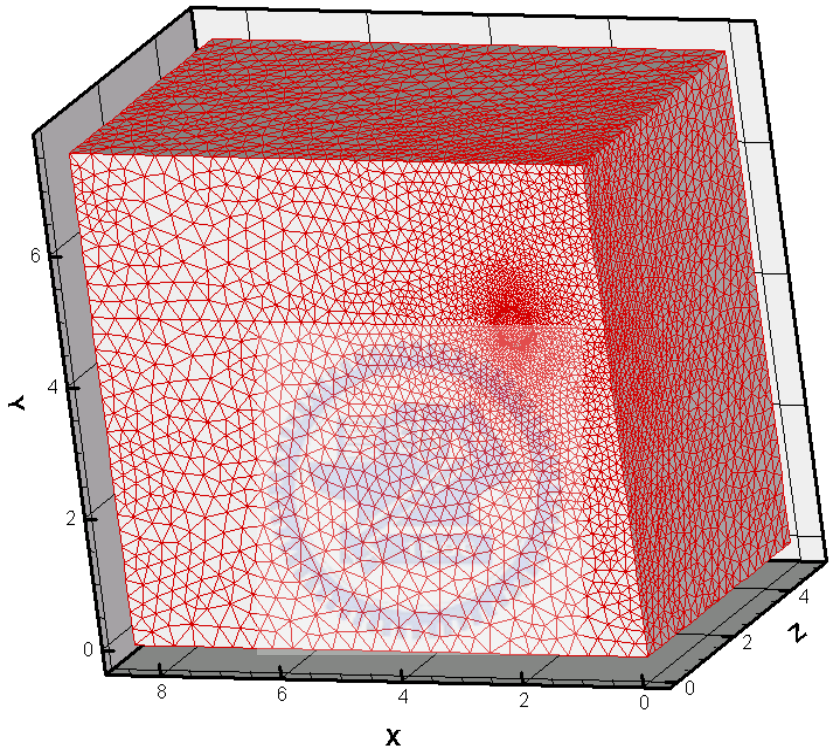
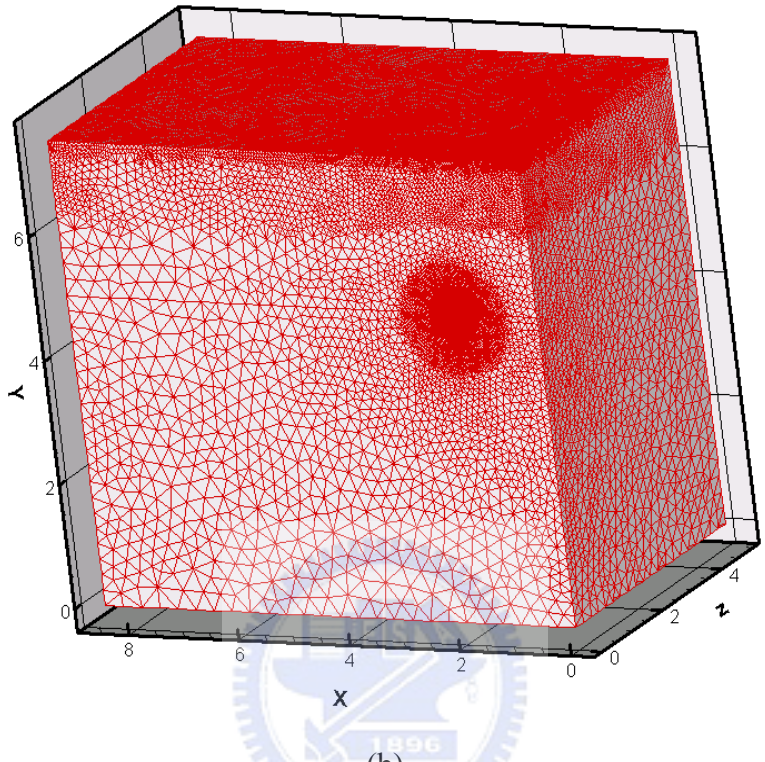


Fig. 4-13 Sketch of two identical charged spheres near a charged infinite flat plate.

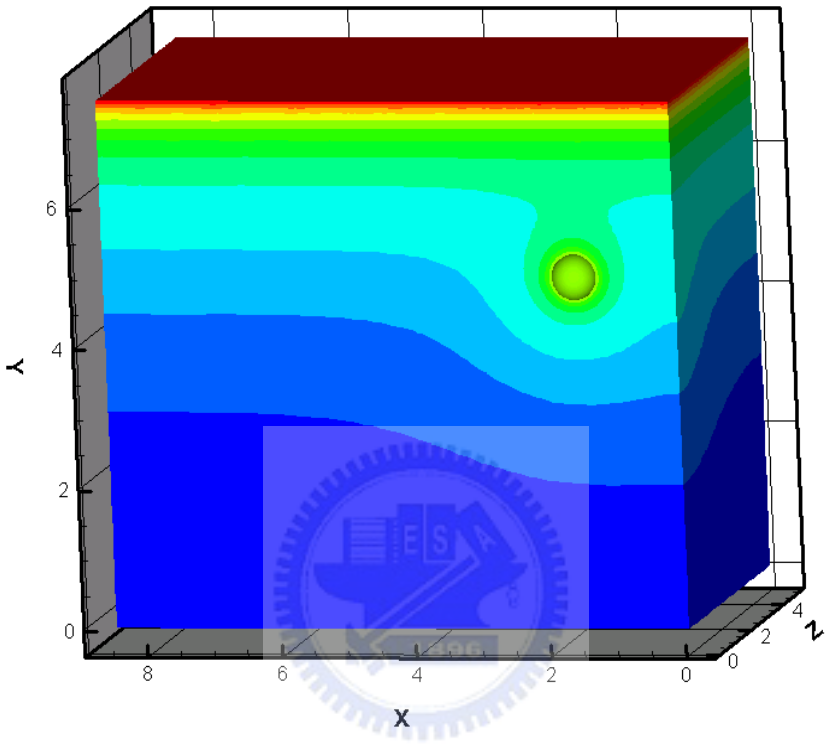


(a)

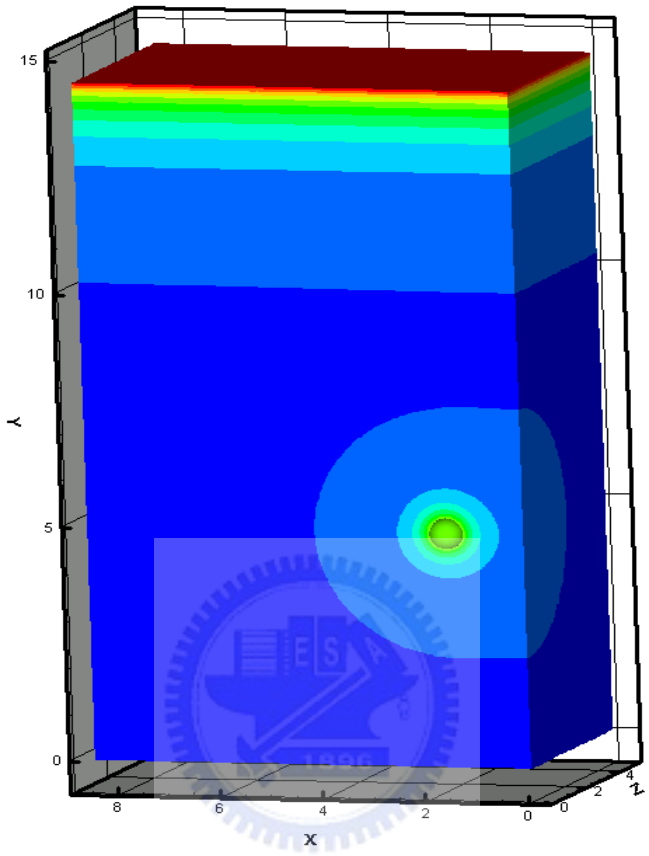


(b)

Fig. 4-14 The surface mesh distribution between two identical charged spheres (radius a=0.325μm) near a like-charged plate (h = 2.5μm, $a/h = 0.13$) at the (a) level-0 (initial, 16,280 nodes; 79,535 elements) (b) level-5 (finial, 185,697 nodes; 1,014,730 elements) mesh refinement.



(a)



(b)

Fig. 4-15 The distribution of potential around the charged spheres (radius a=0.325μm) near a like-charged plate for (a) h = 2.5μm($a/h = 0.13$) (b) h=9.5μm($a/h = 0.03421$).

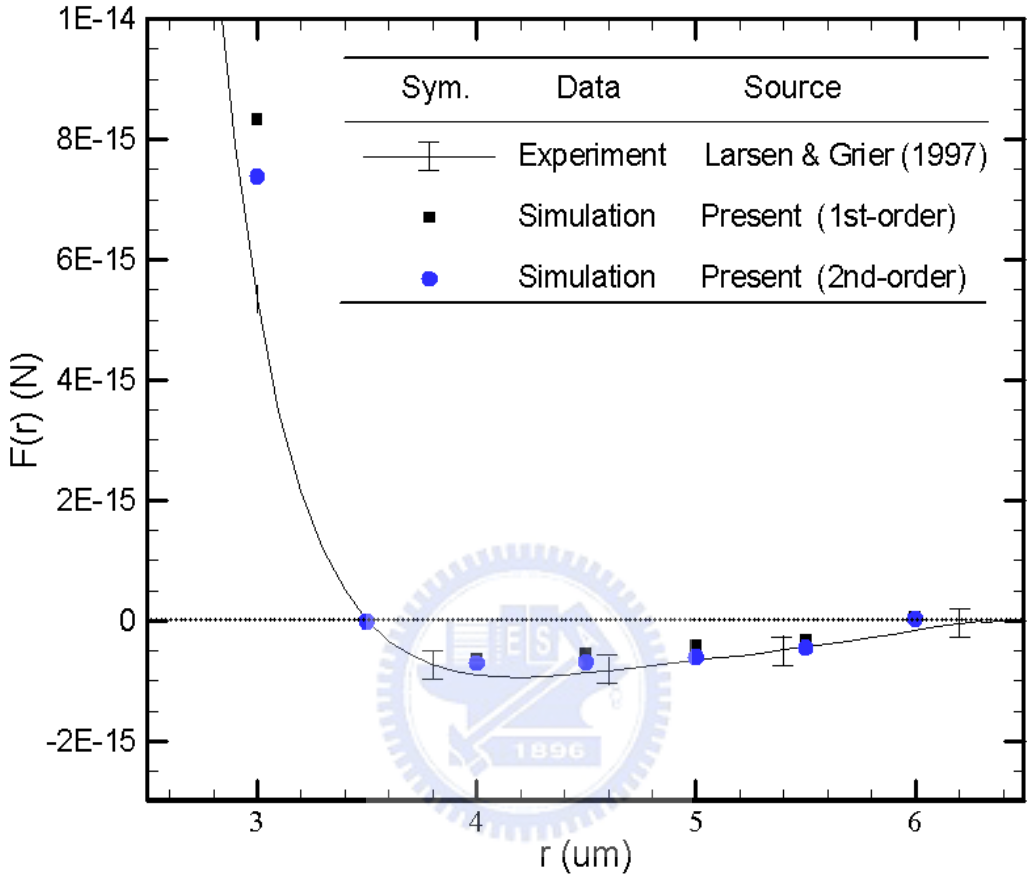


Fig. 4-16 Comparison the first-order shape function with the second-order shape function. Case of computed electrostatic force between charged spheres near a like-charged infinite flat plate as a function of the sphere center-to-center distance (h = 2.5μm, $a/h = 0.13$).

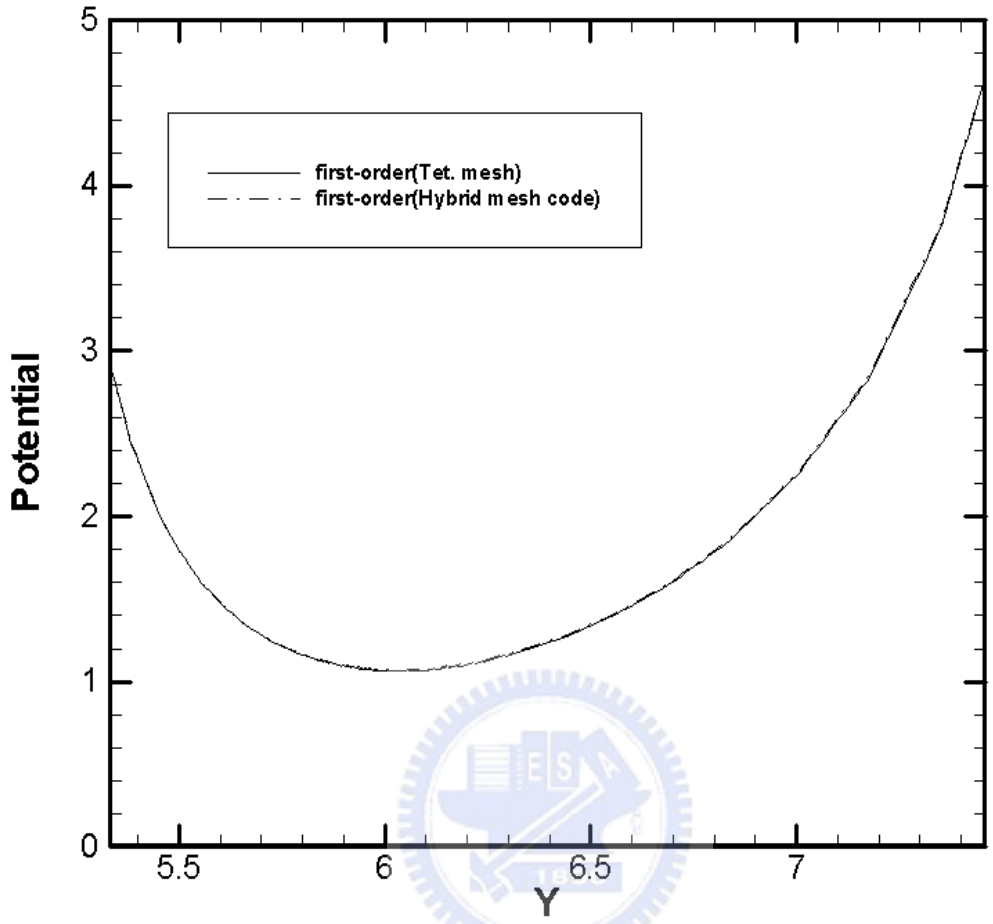


Fig. 5-1 Compare the code of tetrahedral mesh with the code of hybrid mesh of collapsed nodes.

Appendix A Diagonalization of the Jacobian Using Nodal Quadrature

The deductive method of this appendix is provided by Prof. Hwang [web site, http://www.math.ncu.edu.tw/~hwangf/].

In finite element spaces, assume $\mathrm{T}^h = \{K\}$ is a given unstructured mesh with a diameter h_κ. Let $\Psi_0^h \subset H_0^1(\Omega)$ be a trilinear finite element space for the electrical potential ψ:

$$\Psi_0^h = \left\{ \psi \in (C^0(\Omega) \cap H_0^1(\Omega))^3 : \ \psi \mid \kappa \in Q_1(K)^3, K \in T^h \right\} \tag{A-1}$$

which is served as the weighting and trial function space. Here $C^0(\Omega)$ the space consisting of all continuous functions defined on Ω.

Then we can write the finite-dimensional variational formulation as:

Find $\psi^h \in \Psi_0^h$ such that

$$a(\psi^h, \varphi) + (f(\psi^h), \varphi) = 0 \quad \forall \varphi \in \Psi_0^h \tag{A-2}$$

After substituting $\psi^h = \sum_{l=1}^{Nep} \beta_l N_l$ and $\varphi = N_i$ into Galekin formulation, we obtain a system of equation, $F(x) = 0$, where $F = (f_1, f_2, \cdots, f_{Nen})^T$ and $x = (\beta_1, \beta_2, \cdots, \beta_{Nen})^T$ with

$$\begin{aligned} f_i(x) &= a(\sum_{l=1}^{Nep} \beta_l N_l, N_i) + (f(\sum_{l=1}^{Nep} \beta_l N_l), N_i) \\ &= A_K a(\sum_{l=1}^{Nen} \beta_l \widetilde{N}_l, \widetilde{N}_i)_k + A_K (f(\sum_{l=1}^{Nen} \beta_l \widetilde{N}_l), \widetilde{N}_i)_k \end{aligned} \tag{A-3}$$

Next, we use the numerical integration of nodal quadrature.

$$(f(\sum_{l=1}^{N\text{int}}\beta_l\tilde{N}_l),\tilde{N}_i)_k$$

$$=\sum_{m=1}^{N\text{int}}W_m f(\sum_{l=1}^{Nen}\beta_l\tilde{N}_l(\tilde{\xi}_m,\tilde{\eta}_m,\tilde{\varsigma}_m))\tilde{N}_i(\tilde{\xi}_m,\tilde{\eta}_m,\tilde{\varsigma}_m)\hat{j}(\tilde{\xi}_m,\tilde{\eta}_m,\tilde{\varsigma}_m) \qquad \text{(A-4)}$$

Since $\tilde{N}_i(\tilde{\xi}_j,\tilde{\eta}_j,\tilde{\varsigma}_j)=\delta_{ij}$, we have

$$\sum_{m=1}^{N\text{int}}W_m f(\sum_{l=1}^{Nen}\beta_l\delta_{lm})\delta_{im}\hat{j}(\tilde{\xi}_m,\tilde{\eta}_m,\tilde{\varsigma}_m)=W_i f(\beta_i)\hat{j}(\tilde{\xi}_m,\tilde{\eta}_m,\tilde{\varsigma}_m) \qquad \text{(A-5)}$$

For example, 2 x 2 x 2 nodal rules in three dimensions. $N_{\text{int}}=8$, quadrature points:

$\tilde{\xi}_1=-1$, $\tilde{\xi}_2=1$, $\tilde{\xi}_3=1$, $\tilde{\xi}_4=-1$, $\tilde{\xi}_5=-1$, $\tilde{\xi}_6=1$, $\tilde{\xi}_7=1$, $\tilde{\xi}_8=-1$,

$\tilde{\eta}_1=-1$, $\tilde{\eta}_2=-1$, $\tilde{\eta}_3=1$, $\tilde{\eta}_4=1$, $\tilde{\eta}_5=-1$, $\tilde{\eta}_6=-1$, $\tilde{\eta}_7=1$, $\tilde{\eta}_8=1$ and

$\tilde{\zeta}_1=1$, $\tilde{\zeta}_2=1$, $\tilde{\zeta}_3=1$, $\tilde{\zeta}_4=1$, $\tilde{\zeta}_5=-1$, $\tilde{\zeta}_6=-1$, $\tilde{\zeta}_7=-1$, $\tilde{\zeta}_8=-1$

weights $W_i=1$.

Then, we wish to solve

$$\begin{cases} f_1(x_1,x_2)=0 \\ f_2(x_1,x_2)=0 \end{cases} \qquad \text{(A-6)}$$

f_1 and f_2 are nonlinear function of x_1, x_2. Applying Taylor's expansion in two variables around (x_1,x_2) to obtain:

$$\begin{cases} 0=f_1(x_1+h_1,x_1+h_2)\approx f_1(x_1,x_2)+h_1\dfrac{\partial f_1}{\partial x_1}+h_2\dfrac{\partial f_1}{\partial x_2} \\ 0=f_2(x_1+h_1,x_1+h_2)\approx f_2(x_1,x_2)+h_1\dfrac{\partial f_2}{\partial x_1}+h_2\dfrac{\partial f_2}{\partial x_2} \end{cases} \qquad \text{(A-7)}$$

Putting Equation (A-7) into the matrix form

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix} + \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} \tag{A-8}$$

To simplify the notation we introduce the Jacobian matrix

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} \tag{A-9}$$

Then we have

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix} + J \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} \tag{A-10}$$

If J is nonsingular then there exists J^{-1}, so we can solve for $[h_1, h_2]^T$.

$$\begin{bmatrix} h_1 \\ h_2 \end{bmatrix} = -J^{-1} \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix} \tag{A-11}$$

For Newton's method

$$\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix} + \begin{bmatrix} h_1^{(k)} \\ h_2^{(k)} \end{bmatrix} \tag{A-12}$$

With

$$J \begin{bmatrix} h_1^{(k)} \\ h_2^{(k)} \end{bmatrix} = - \begin{bmatrix} f_1(x_1^{(k)}, x_2^{(k)}) \\ f_2(x_1^{(k)}, x_2^{(k)}) \end{bmatrix} \tag{A-13}$$

In general, we can use Newton's method for $F(x) = 0$, where

$x = (\beta_1, \beta_2, ..., \beta_n)^T$ and $F = (f_1, f_2, ..., f_n)^T$. For higher dimensional function, the first derivative is defined as a matrix

$$F'(x)=\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{A-14}$$

Then, calculation of Jacobian for the PB equation, write the Jacobian matrices corresponding to in the form of $J = J^L + J^N$, where $J^N = (J_{ij}^N) = \sum_k (J_{ij}^N)_k$, with nodal quadrature

$$(J_{ij}^N)_k = \begin{cases} \frac{\partial f(\beta_i)}{\partial \beta_j} = W_i f(\beta_i) \tilde{j}(\tilde{\xi}_i, \tilde{\eta}_i, \tilde{\zeta}_i), & i = j \\ 0, & i \neq j \end{cases} \tag{A-15}$$

We consider the Jacobian system

$$J(x^{(n)})s^{(n)} = -F(x^{(n)}) \tag{A-16}$$

Taking the full Newton step, i.e. $s^{(n)} = x^{(n+1)} - x^{(n)}$ to yield

$$0 = F(x^{(n)}) + (x^{(n+1)} - x^{(n)})J(x^{(n)}) \tag{A-17}$$

or

$$J(x^{(n)})x^{(n+1)} = J(x^{(n)})x^{(n)} - F(x^{(n)}) \tag{A-18}$$

Since $J(x^{(n)}) = J^L + J^N(x^{(n)})$ and $F(x^{(n)}) = J^L x^{(n)} - G(x^{(n)})$, we have

$$(J^L + J^N(x^{(n)}))x^{(n+1)} = (J^L + J^N(x^{(n)}))x^{(n)} - (J^L x^{(n)} + G(x^{(x)})) \tag{A-19}$$

Or

$$(J^L + J^N(x^{(n)}))x^{(n+1)} = (J^N(x^{(n)}))x^{(n)} - G(x^{(x)}) \tag{A-20}$$

Then the new approximation $U^{(k+1)}$ can be computed through

$$A^{(k)}U^{(k+1)} = B(U^{(k)}), \tag{A-21}$$

where $A^{(k)} = (J^L + J^N(x^{(n)}))$ and $B(U^{(k)}) = (J^N(x^{(n)}))x^{(n)} - G(x^{(x)})$. In our implementation, the non-zero entries of $A^{(k)}$ are stored in the compressed sparse row (CSR) format that is computationally efficient both in terms of storage and matrix operations.

Appendix B Three-dimensional Hybrid Mesh

B-1 Hybrid Mesh

Through tetrahedron element are used for many simulations. In some cases the use of pyramid, wedge or hexahedral elements may be attractive. A nature way of generating tetrahedron element appears to be to simple distort the basic hexahedral element into the required collapsing form. This is achieved in practice by assigning the same global node to four corner nodes of element which from hexahedral element to pyramid element. If we collapse another node to two corner nodes of element further. This element will be a tetrahedron element from pyramid element, shown in Figure B-1.

B-2 Shape Function of Hexahedral Element

The simplest element in this family is the eight-node trilinear hexahedron, or brick, show in Figure B-2.

The interpolation function is

$$\phi = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 z + \alpha_5 xy + \alpha_6 xz + \alpha_7 yz + \alpha_8 xyz \tag{B-1}$$

Equation (B-1) can be expressed in matrix form as

$$\begin{bmatrix} \phi_1 \\ \vdots \\ \phi_8 \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & z_1 & x_1y_1 & x_1z_1 & y_1z_1 & x_1y_1z_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_8 & y_8 & z_8 & x_8y_8 & x_8z_8 & y_8z_8 & x_8y_8z_8 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_8 \end{bmatrix} \tag{B-2}$$

Since $\alpha = C^{-1}\phi$, we can find the values for ϕ from the expression

$$\phi = [1 \;\; x \;\; y \;\; z \;\; xy \;\; xz \;\; yz \;\; xyz]C^{-1}\phi \tag{B-3}$$

from which the shape functions are obtained as before:

$$N = [1 \;\; x \;\; y \;\; z \;\; xy \;\; xz \;\; yz \;\; xyz]C^{-1} \tag{B-4}$$

The shape functions are defined as

$$\begin{cases} N_1(\xi,\eta,\rho) = \frac{1}{8}(1-\xi)(1-\eta)(1-\rho) \\ N_2(\xi,\eta,\rho) = \frac{1}{8}(1+\xi)(1-\eta)(1-\rho) \\ N_3(\xi,\eta,\rho) = \frac{1}{8}(1+\xi)(1+\eta)(1-\rho) \\ N_4(\xi,\eta,\rho) = \frac{1}{8}(1-\xi)(1+\eta)(1-\rho) \\ N_5(\xi,\eta,\rho) = \frac{1}{8}(1-\xi)(1-\eta)(1+\rho) \\ N_6(\xi,\eta,\rho) = \frac{1}{8}(1+\xi)(1-\eta)(1+\rho) \\ N_7(\xi,\eta,\rho) = \frac{1}{8}(1+\xi)(1+\eta)(1+\rho) \\ N_8(\xi,\eta,\rho) = \frac{1}{8}(1-\xi)(1+\eta)(1+\rho) \end{cases} \tag{B-5}$$

or $N_i(\xi,\eta,\rho) = \frac{1}{8}(1+\xi\xi_i)(1+\eta\eta_i)(1+\rho\rho_i)$. ξ_i, η_i and ρ_i are +1 or -1, depend on the direction of the node on the local coordinate.

In an element, the differential shape function with ξ, η or ρ is the same as other elements. The Jacobian is given by

$$[J] = \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} & \frac{\partial N_3}{\partial \xi} & \frac{\partial N_4}{\partial \xi} & \frac{\partial N_5}{\partial \xi} & \frac{\partial N_6}{\partial \xi} & \frac{\partial N_7}{\partial \xi} & \frac{\partial N_8}{\partial \xi} \\ \frac{\partial N_1}{\partial \eta} & \frac{\partial N_2}{\partial \eta} & \frac{\partial N_3}{\partial \eta} & \frac{\partial N_4}{\partial \eta} & \frac{\partial N_5}{\partial \eta} & \frac{\partial N_6}{\partial \eta} & \frac{\partial N_7}{\partial \eta} & \frac{\partial N_8}{\partial \eta} \\ \frac{\partial N_1}{\partial \rho} & \frac{\partial N_2}{\partial \rho} & \frac{\partial N_3}{\partial \rho} & \frac{\partial N_4}{\partial \rho} & \frac{\partial N_5}{\partial \rho} & \frac{\partial N_6}{\partial \rho} & \frac{\partial N_7}{\partial \rho} & \frac{\partial N_8}{\partial \rho} \end{bmatrix} \begin{bmatrix} X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \\ X_3 & Y_3 & Z_3 \\ X_4 & Y_4 & Z_4 \\ X_5 & Y_5 & Z_5 \\ X_6 & Y_6 & Z_6 \\ X_7 & Y_7 & Z_7 \\ X_8 & Y_8 & Z_8 \end{bmatrix}$$

$$= \frac{1}{8}\begin{bmatrix} -(1-\eta)(1-\rho) & +(1-\eta)(1-\rho) & +(1+\eta)(1-\rho) & -(1+\eta)(1-\rho) & , \\ -(1-\xi)(1-\rho) & -(1+\xi)(1-\rho) & +(1+\xi)(1-\rho) & +(1-\xi)(1-\rho) & , \\ -(1-\xi)(1-\eta) & -(1+\xi)(1-\eta) & -(1+\xi)(1+\eta) & -(1-\xi)(1+\eta) & , \end{bmatrix}$$

$$\begin{bmatrix} -(1-\eta)(1+\rho) & +(1-\eta)(1+\rho) & +(1+\eta)(1+\rho) & -(1+\eta)(1+\rho) \\ -(1-\xi)(1+\rho) & -(1+\xi)(1+\rho) & +(1+\xi)(1+\rho) & +(1-\xi)(1+\rho) \\ +(1-\xi)(1-\eta) & +(1+\xi)(1-\eta) & +(1+\xi)(1+\eta) & +(1-\xi)(1+\eta) \end{bmatrix}$$

$$\begin{pmatrix} X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \\ X_3 & Y_3 & Z_3 \\ X_4 & Y_4 & Z_4 \\ X_5 & Y_5 & Z_5 \\ X_6 & Y_6 & Z_6 \\ X_7 & Y_7 & Z_7 \\ X_8 & Y_8 & Z_8 \end{pmatrix}$$

$$= \frac{1}{8}\begin{bmatrix} \sum_{i=1}^{8}\frac{\partial N_i}{\partial \xi}X_i & \sum_{i=1}^{8}\frac{\partial N_i}{\partial \xi}Y_i & \sum_{i=1}^{8}\frac{\partial N_i}{\partial \xi}Z_i \\ \sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}X_i & \sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}Y_i & \sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}Z_i \\ \sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}X_i & \sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}Y_i & \sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}Z_i \end{bmatrix} = \frac{1}{8}[C] \qquad \text{(B-6)}$$

and $\left|\det[J]\right| = \frac{1}{8^3}|C|$. The transpose is

$$[J]^T = \frac{1}{8}\begin{bmatrix} \sum_{i=1}^{8}\frac{\partial N_i}{\partial \xi}X_i & \sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}X_i & \sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}X_i \\ \sum_{i=1}^{8}\frac{\partial N_i}{\partial \zeta}Y_i & \sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}Y_i & \sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}Y_i \\ \sum_{i=1}^{8}\frac{\partial N_i}{\partial \zeta}Z_i & \sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}Z_i & \sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}Z_i \end{bmatrix} \qquad \text{(B-7)}$$

The adjoint is

$$adj([J]) = \frac{1}{8}\begin{bmatrix} +(\sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}Y_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}Z_i - \sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}Z_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}Y_i) & , \\ -(\sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}X_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}Z_i - \sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}Z_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}X_i) & , \\ +(\sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}X_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}Y_i - \sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}Y_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}X_i) & , \\ -(\sum_{i=1}^{8}\frac{\partial N_i}{\partial \xi}Y_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}Z_i - \sum_{i=1}^{8}\frac{\partial N_i}{\partial \xi}Z_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}Y_i) & , \\ +(\sum_{i=1}^{8}\frac{\partial N_i}{\partial \xi}X_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}Z_i - \sum_{i=1}^{8}\frac{\partial N_i}{\partial \xi}Z_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}X_i) & , \\ -(\sum_{i=1}^{8}\frac{\partial N_i}{\partial \xi}X_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}Y_i - \sum_{i=1}^{8}\frac{\partial N_i}{\partial \xi}Y_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \rho}X_i) & , \\ +(\sum_{i=1}^{8}\frac{\partial N_i}{\partial \xi}Y_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}Z_i - \sum_{i=1}^{8}\frac{\partial N_i}{\partial \xi}Z_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}Y_i) & \\ -(\sum_{i=1}^{8}\frac{\partial N_i}{\partial \xi}X_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}Z_i - \sum_{i=1}^{8}\frac{\partial N_i}{\partial \xi}Z_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}X_i) & \\ +(\sum_{i=1}^{8}\frac{\partial N_i}{\partial \xi}X_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}Y_i - \sum_{i=1}^{8}\frac{\partial N_i}{\partial \xi}Y_i\sum_{i=1}^{8}\frac{\partial N_i}{\partial \eta}X_i) & \end{bmatrix} \quad \text{(B-8)}$$

The inverse is

$$[J]^{-1} = \frac{adj([J])}{\det([J])} \quad \text{(B-9)}$$

Hence,

$$\begin{bmatrix} \frac{\partial N_k(\xi,\eta,\rho)}{\partial X} \\ \frac{\partial N_k(\xi,\eta,\rho)}{\partial Y} \\ \frac{\partial N_k(\xi,\eta,\rho)}{\partial Z} \end{bmatrix} = [J^{-1}]\begin{bmatrix} \frac{\partial N_k(\xi,\eta,\rho)}{\partial \xi} \\ \frac{\partial N_k(\xi,\eta,\rho)}{\partial \eta} \\ \frac{\partial N_k(\xi,\eta,\rho)}{\partial \rho} \end{bmatrix} = \frac{1}{8}\frac{adj([J])}{|\det(|J|)|}\begin{bmatrix} \xi_k(1+\eta\eta_k)(1+\rho\rho_k) \\ \eta_k(1+\xi\xi_k)(1+\rho\rho_k) \\ \rho_k(1+\xi\xi_k)(1+\eta\eta_k) \end{bmatrix} \quad \text{(B-10)}$$

We use the results of differential shape function with X or Y in substitution for the one of Poisson equation.

$$\sum\sum\sum\iiint\left[(\frac{\partial N_{k1}}{\partial X}\frac{\partial N_{k2}}{\partial X}) + (\frac{\partial N_{k1}}{\partial Y}\frac{\partial N_{k2}}{\partial Y}) + (\frac{\partial N_{k1}}{\partial Z}\frac{\partial N_{k2}}{\partial Z})\right]a_j\, dxdydz \quad \text{(B-11)}$$

Finally, we use the Gauss points and weights to perform the numerical integration.

B-3 The Second-Order shape function

Extension of the 8-noded quadratic element to the 20-noded quadratic hexahedron can be easily made by adding a mid-side node to each edge of the element. The 3D quadratic hexahedron is shown in Fig B-3.

We can use the same as processes to degenerate the three-dimensional hexahedron element of 20-noded to wedge, pyramid, tetrahedral mesh shown in Fig B-4. There are 27 Gauss points to perform the numerical integration.

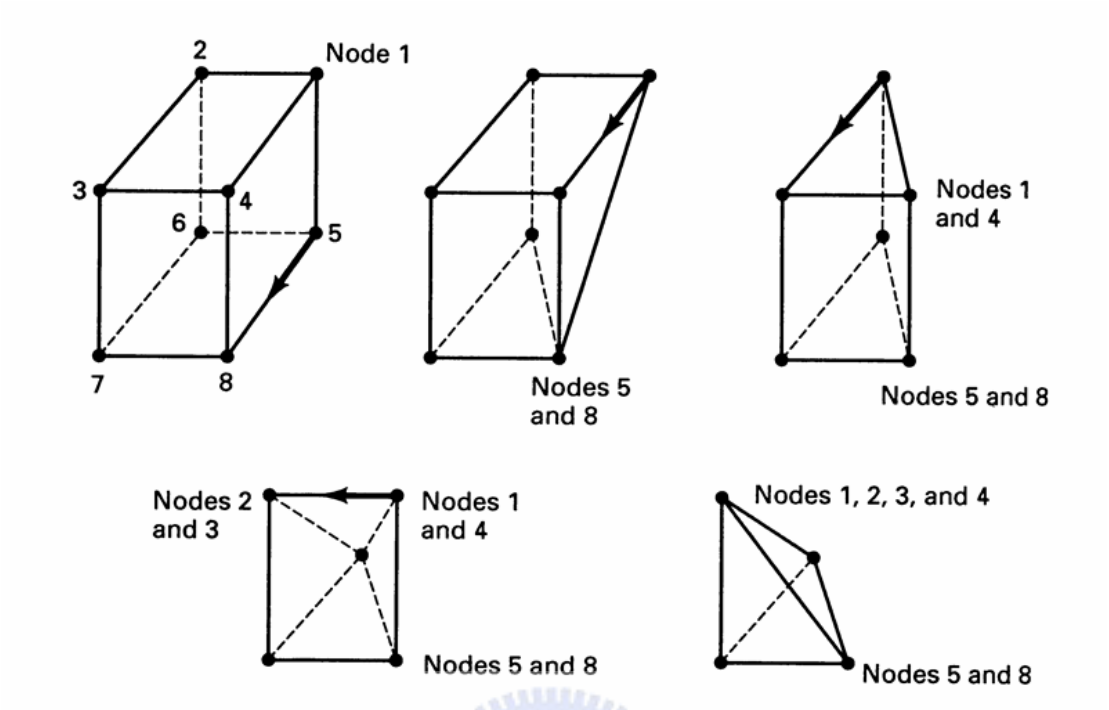


Fig. B-1 Degenerate forms of 8-node three-dimensional element

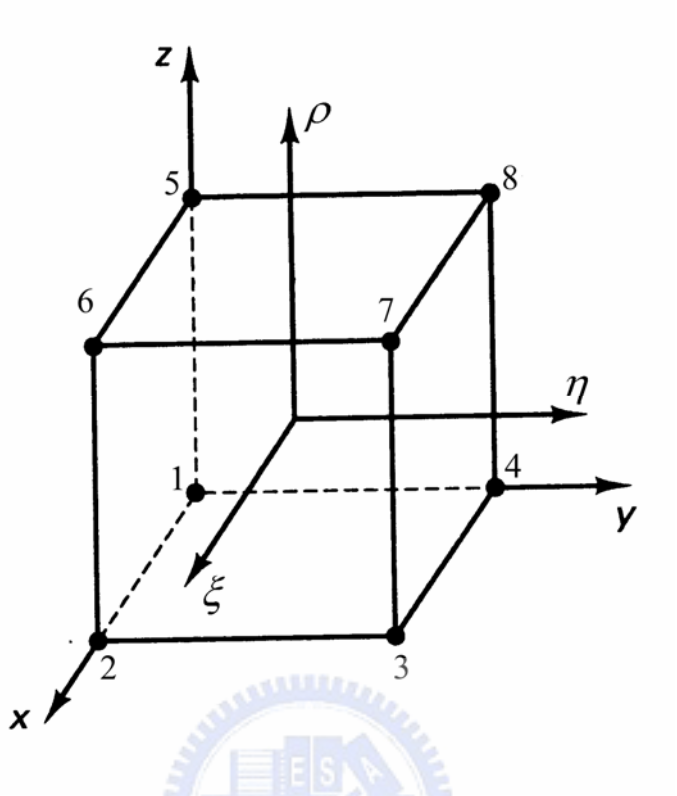


Fig. B-2 the eight-node trilinear hexahedron element

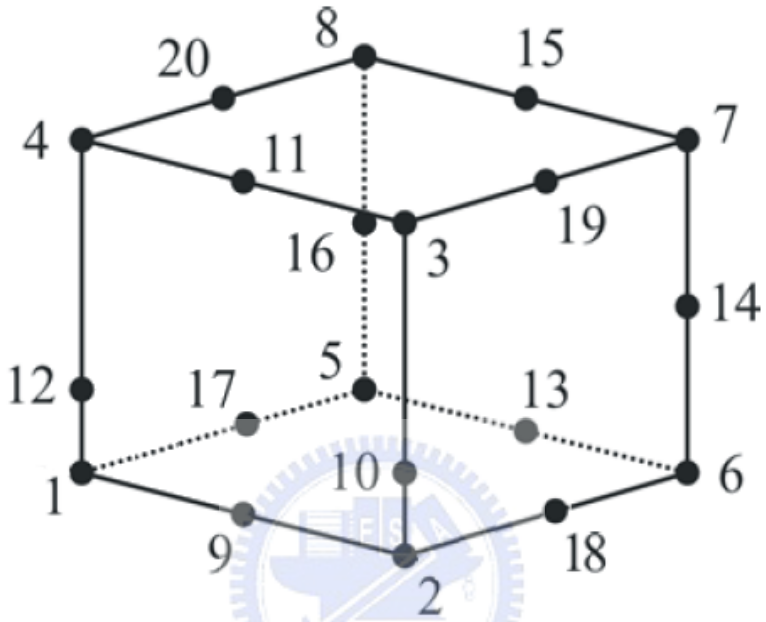


Fig. B-3 The 3D quadratic hexahedron

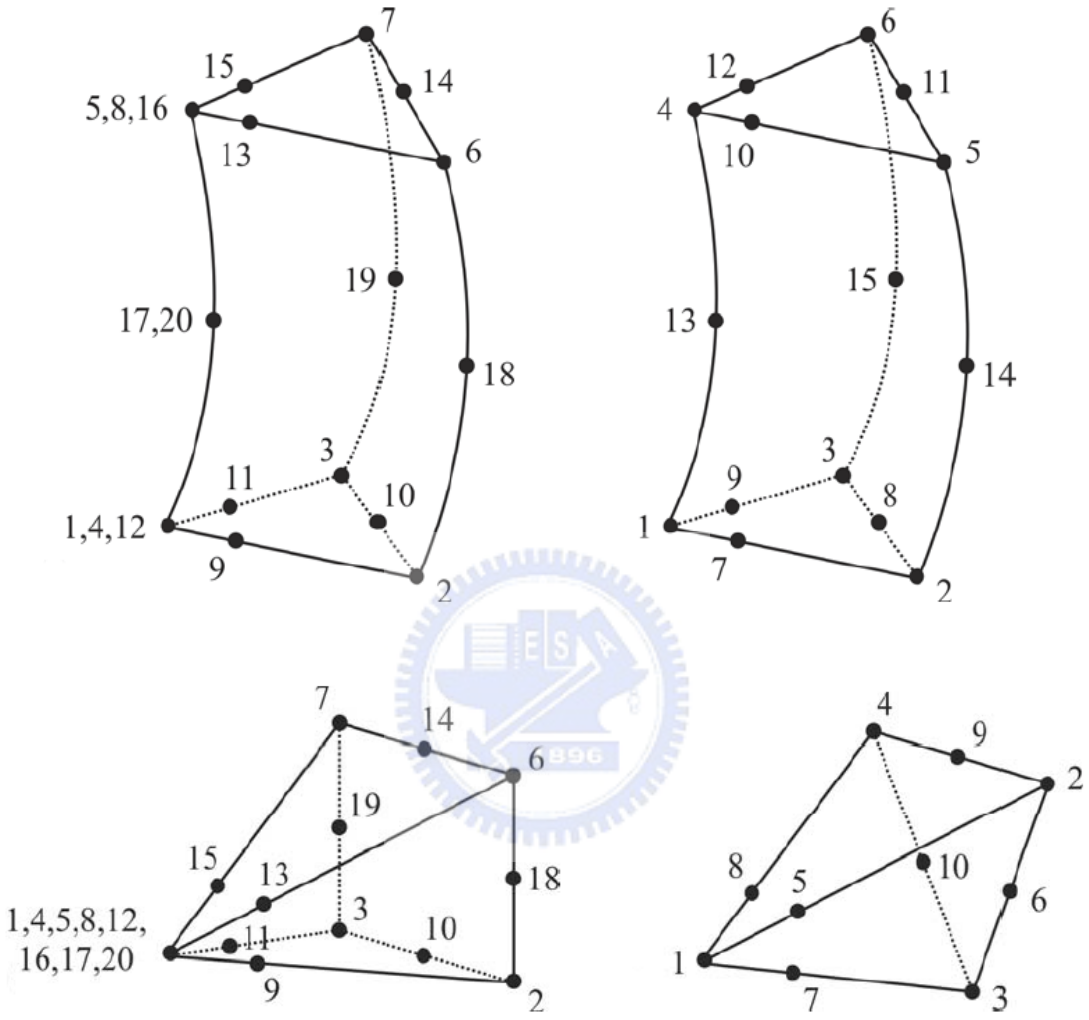


Fig. B-4 Degenerate forms of 20-node three-dimensional element

Appendix C First-order Shape Function of Tetrahedron Element

Four coordinates L_1, L_2, L_3 and L_4 will be used to define a point P, although only three of them are independent. There natural coordinates are defined as

$$L_1 = \frac{V_1}{V},\quad L_2 = \frac{V_2}{V},\quad L_3 = \frac{V_3}{V} \quad \text{and} \quad L_4 = \frac{V_4}{V} \tag{C-1}$$

where V_i is the volume of the tetrahedron formed by the point P and the vertices other than vertex i, and V is the volume of tetrahedron element defined by vertices 1, 2, 3 and 4 (Fig. C-1). The volume of tetrahedron element by the point P are defined as

$$V_1 = Volume\ of\ P234,\quad V_2 = Volume\ of\ P134,$$

$$V_3 = Volume\ of\ P124 \quad \text{and} \quad V_4 = Volume\ of\ P123 \tag{C-2}$$

Because the natural coordinates are defined in terms of volumes, they are also known as volume or tetrahedral coordinates. Since $\sum_{i=1}^{4} V_i = V$, we obtain

$$\frac{V_1}{V} + \frac{V_2}{V} + \frac{V_3}{V} + \frac{V_4}{V} = L_1 + L_2 + L_3 + L_4 = 1 \tag{C-3}$$

The volume coordinates L_i are also the shape functions for a three dimensional simplex element.

The Cartesian and natural coordinates are related as

$$\begin{cases} x = L_1 x_1 + L_2 x_2 + L_3 x_3 + L_4 x_4 \\ y = L_1 y_1 + L_2 y_2 + L_3 y_3 + L_4 y_4 \\ z = L_1 z_1 + L_2 z_2 + L_3 z_3 + L_4 z_4 \end{cases} \tag{C-4}$$

Equation (C-3) and (C-4) can be expressed in matrix form as

$$\begin{bmatrix} 1 \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \end{bmatrix} \tag{C-5}$$

The inverse relations can be expressed as

$$\begin{bmatrix} L_1 \\ L_2 \\ L_3 \\ L_4 \end{bmatrix} = \frac{1}{6V} \begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \\ a_4 & b_4 & c_4 & d_4 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \\ z \end{bmatrix} \tag{C-6}$$

where

$$a_1 = \begin{vmatrix} x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \end{vmatrix}, \quad b_1 = -\begin{vmatrix} 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \\ 1 & y_4 & z_4 \end{vmatrix}, \quad c_1 = -\begin{vmatrix} x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \\ x_4 & 1 & z_4 \end{vmatrix} \quad \text{and} \quad d_1 = -\begin{vmatrix} x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \end{vmatrix}$$

Other constants are obtained through a cyclic permutation of subscripts 1, 2, 3 and 4.

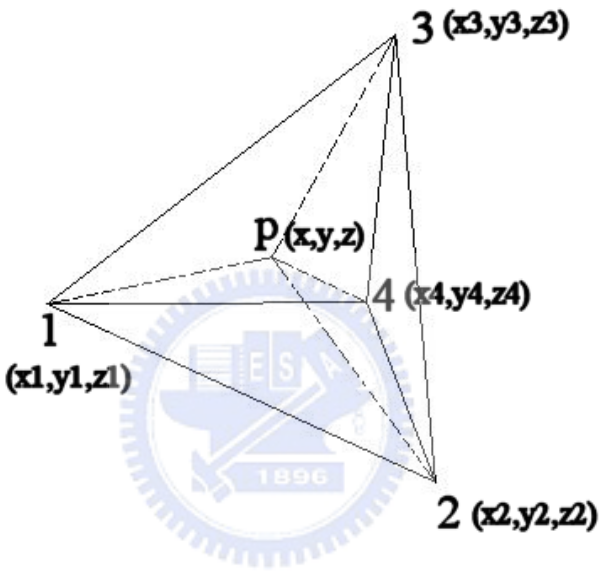


Fig. C-1 Volume coordinates for a tetrahedron element.

Appendix D Second-order Shape Function of Tetrahedron Element

In this appendix, we provided some implementation of the matrix of coefficients for the second-order shape function of tetrahedron element.

The volume integrals can be easily evaluated from the relation

$$\int_V L_1^a L_2^b L_3^c L_4^d dv = \frac{a!\,b!\;\;c!\,d!}{(3+a+b+c+d)!}6V \tag{D-1}$$

We can use the equation to develop the volume integral, then

$$\begin{aligned}\nabla\phi_1 &= \frac{1}{18V^2}\begin{Bmatrix}[(2b_1^{*2}x)-3V\cdot b_1^*+2a_1^*b_1^*+2b_1^*c_1^*y+2b_1^*d_1^*z]\vec{i}\\ +[(2c_1^{*2}y)-3V\cdot c_1^*+2a_1^*c_1^*+2b_1^*c_1^*y+2c_1^*d_1^*z]\vec{j}\\ +[(2d_1^{*2}z)-3V\cdot d_1^*+2a_1^*d_1^*+2b_1^*d_1^*y+2c_1^*d_1^*y]\vec{k}\end{Bmatrix}\\ &= \frac{1}{3V}\begin{Bmatrix}b_1^*[-\frac{1}{2}+2L_1]\vec{i}\\ +c_1^*[-\frac{1}{2}+2L_1]\vec{j}\\ +d_1^*[-\frac{1}{2}+2L_1]\vec{k}\end{Bmatrix}\end{aligned} \tag{D-2}$$

$$\begin{aligned}\nabla\phi_1\cdot\nabla\phi_1 &= \frac{(b_1^{*2}+c_1^{*2}+d_1^{*2})(-\frac{1}{2}+2L_1)^2}{9v^2}\\ &= \frac{(b_1^{*2}+c_1^{*2}+d_1^{*2})}{9v^2}(4L_1^2-2L_1+\frac{1}{4})\end{aligned} \tag{D-4}$$

$$\iiint\nabla\phi_1\cdot\nabla\phi_1 dxdydz = \frac{(b_1^{*2}+c_1^{*2}+d_1^{*2})}{9v}\frac{3}{20} \tag{D-5}$$

For $i \neq j, i, j \leq 4$,

$$\nabla\phi_i \cdot \nabla\phi_j = \frac{(b_i^* b_j^* + c_i^* c_j^* + d_i^* d_j^*)(-\frac{1}{2} + 2L_i)(-\frac{1}{2} + 2L_j)}{9v^2} = \frac{(b_i^* b_j^* + c_i^* c_j^* + d_i^* d_j^*)}{9v^2}(4L_i L_j - L_i - L_j + \frac{1}{4}) \tag{D-6}$$

$$\iiint \nabla\phi_i \cdot \nabla\phi_j dxdydz = \frac{(b_i^* b_j^* + c_i^* c_j^* + d_i^* d_j^*)}{9v}\frac{-1}{20} \tag{D-7}$$

$$\begin{aligned}\nabla\phi_1 \cdot \nabla\phi_5 &= \frac{2}{9v^2}(-\frac{1}{2} + 2L_1)(b_1^*(b_1^* L_2 + b_2^* L_1) + c_1^*(c_1^* L_2 + c_2^* L_1) \\ &\qquad + d_1^*(d_1^* L_2 + d_2^* L_1)) \\ &= \frac{2}{9v^2}(-\frac{1}{2} + 2L_1)(L_1(b_1^* b_2^* + c_1^* c_2^* + d_1^* d_2^*) + L_2(b_1^{*2} + c_1^{*2} + d_1^{*2}))\end{aligned} \tag{D-8}$$

then, to use the equation to develop the volume integral.

$$\begin{aligned}\nabla\phi_1 \cdot \nabla\phi_8 &= \frac{2}{9v^2}(-\frac{1}{2} + 2L_1)(b_1^*(b_2^* L_3 + b_3^* L_2) + c_1^*(c_2^* L_3 + c_3^* L_2) \\ &\qquad + d_1^*(d_2^* L_3 + d_3^* L_2)) \\ &= \frac{2}{9v^2}(2L_1 L_2(b_1^* b_3^* + c_1^* c_3^* + d_1^* d_3^*) + 2L_1 L_3(b_1^* b_2^* + c_1^* c_2^* + d_1^* d_2^*) \\ &\quad -\frac{1}{2}L_2(b_1^* b_3^* + c_1^* c_3^* + d_1^* d_3^*) - \frac{1}{2}L_3(b_1^* b_2^* + c_1^* c_2^* + d_1^* d_2^*))\end{aligned} \tag{D-9}$$

For $i, j > 4$,

$$\nabla\phi_{i,j} = \frac{1}{9V^2}\begin{Bmatrix}[2b_i^* b_j^* x + (a_i^* b_j^* + b_i^* a_j^*)x + (b_i^* c_j^* + c_i^* b_j^*)y + (d_i^* b_j^* + b_i^* d_j^*)z]\vec{i} \\ +[2c_i^* c_j^* y + (a_i^* c_j^* + c_i^* a_j^*)x + (b_i^* c_j^* + c_i^* b_j^*)y + (c_i^* d_j^* + d_i^* c_j^*)z]\vec{j} \\ +[2d_i^* d_j^* z + (a_i^* d_j^* + d_i^* a_j^*)x + (b_i^* d_j^* + d_i^* b_j^*)y + (c_i^* d_j^* + d_i^* c_j^*)z]\vec{k}\end{Bmatrix} \tag{D-10}$$

$$= \frac{2}{3V}\begin{Bmatrix}[b_i^* L_j + b_j^* L_i]\vec{i} \\ +[c_i^* L_j + c_j^* L_i]\vec{j} \\ +[d_i^* L_j + d_j^* L_i]\vec{k}\end{Bmatrix}$$

$$\nabla\phi_5\cdot\nabla\phi_5=\frac{1}{81V^4}\left\{\begin{array}{l}[b_2^*(a_1^*+b_1^*x+c_1^*y+d_1^*z)+b_1^*(a_2^*+b_2^*x+c_2^*y+d_2^*z)]^2\\+[c_2^*(a_1^*+b_1^*x+c_1^*y+d_1^*z)+c_1^*(a_2^*+b_2^*x+c_2^*y+d_2^*z)]^2\\+[d_2^*(a_1^*+b_1^*x+c_1^*y+d_1^*z)+d_1^*(a_2^*+b_2^*x+c_2^*y+d_2^*z)]^2\end{array}\right\} \quad \text{(D-11)}$$

$$=\frac{4}{9V^2}[(b_1^*L_2+b_2^*L_1)^2+(c_1^*L_2+c_2^*L_1)^2+(d_1^*L_2+d_2^*L_1)^2]$$

therefore, when $i=j,\ i,j>4$,

$$\iiint\nabla\phi_i\cdot\nabla\phi_j dxdydz=\frac{4}{90v}[(b_1^{*2}+c_1^{*2}+d_1^{*2})+(b_2^{*2}+c_2^{*2}+d_2^{*2}) +(b_1^*b_2^*+c_1^*c_2^*+d_1^*d_2^*)] \quad \text{(D-12)}$$

When $i\neq j,\ i,j>4$,

$$\nabla\phi_5\cdot\nabla\phi_6=\frac{4}{9v^2}[(b_1^*L_2+b_2^*L_1)(b_1^*L_3+b_3^*L_1)+(c_1^*L_2+c_2^*L_1)(c_1^*L_3+c_3^*L_1) +(d_1^*L_2+d_2^*L_1)(d_1^*L_3+d_3^*L_1)] \quad \text{(D-13)}$$

$$=\frac{4}{9v^2}[L_2L_3(b_1^{*2}+c_1^{*2}+d_1^{*2})+L_1L_2(b_1^*b_3^*+c_1^*c_3^*+d_1^*d_3^*) +L_1L_3(b_1^*b_2^*+c_1^*c_2^*+d_1^*d_2^*)+L_1L_1(b_2^*b_3^*+c_2^*c_3^*+d_2^*d_3^*)]$$

$$\nabla\phi_5\cdot\nabla\phi_{10}=\frac{4}{9v^2}[(b_1^*L_2+b_2^*L_1)(b_3^*L_4+b_4^*L_3)+(c_1^*L_2+c_2^*L_1)(c_3^*L_4+c_4^*L_3) +(d_1^*L_2+d_2^*L_1)(d_3^*L_4+d_4^*L_3)] \quad \text{(D-14)}$$

$$=\frac{4}{9v^2}[L_2L_3(b_1^*b_4^*+c_1^*c_4^*+d_1^*d_4^*)+L_2L_4(b_1^*b_3^*+c_1^*c_3^*+d_1^*d_3^*) +L_1L_4(b_2^*b_3^*+c_2^*c_3^*+d_2^*d_3^*)+L_1L_3(b_2^*b_4^*+c_2^*c_4^*+d_2^*d_4^*)]$$

Autobiography

Yun-Long Shao
Phone: 03-5712121-55175
Email: maty.me89g@nctu.edu.tw

Interests and Objectives

1. **Scalable distributed computing**: parallel computing using finite-element method (FEM) and Lattice Boltzmann method (LBM) with domain decomposition and other related parallel high-performance computation.
2. **Computational bio-physics**: a parallel PB equation solver with parallel adaptive mesh refinement (PAMR), which models electrical double layer around charged sphere or a cylinder.
3. **Microfluidics**: familiar in microfabrication of microfluidics devices; micromixing, micropumping.

Education

2000-Present	PhD program of M.E., National Chiao-Tung University
1996-1998	M. S. in Mechanical Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan MS Thesis: Dynamic Simulation and Heat Transfer Analysis of Reheating Furnace
1993-1996	B. S. in Mechanical Engineering, Yuan-Ze University, Taoyuan, Taiwan
1985-1990	Associate Degree in Mechanical Engineering, Far East College, Tainan, Taiwan

Languages

Mandarin Chinese, Taiwanese and English

Advisor

Professor J.-S. Wu
Department of Mechanical Engineering
National Chiao-Tung University
Email: chongsin@cc.nctu.edu.tw

List of Publications

Referred Papers (* represent work directly related to PhD thesis):

01. J.-S. Wu and **Y.-L. Shao**, "Simulation of Lid-Driven Cavity Flows by Parallel Lattice Boltzmann Method Using Multi-Relaxation Time Scheme," International Journal for Numerical Methods in Fluids, Vol. 46, pp. 921-937, 2004. (SCI)
02. J.-S. Wu, H.-Y. Chou, U.-M. Lee, **Y.-L. Shao** and Y.-Y. Lian, "Parallel DSMC Simulation of a Single Underexpanded Free Jet from Transition to Near-continuum Regime," ASME-Journal of Fluids Engineering, Vol.127, pp.1161-1170, 2005. (SCI)
03. J.-S. Wu and **Y.-L. Shao**, "Simulation of Flow Past a Square Cylinder by Parallel Lattice Boltzmann Method Using Multi-Relaxation-Time Scheme," Journal of Mechanics, Vol. 22, NO. 1, pp.35-42, 2006. (SCI)
04. Y.-Y. Lian, K.-H. Hsu, **Y.-L. Shao**, Y.-M. Lee, Y.-W. Jeng and J.-S. Wu, "Parallel Adaptive Mesh-Refining Scheme on Three-dimensional Unstructured Tetrahedral Mesh and Its Applications," Computer Physics Communications, 2006. (SCI, accepted)*
05. **Y.-L. Shao**, K.-H. Hsu, J.-S. Wu, Y.-Y. Lian and F.-N. Hwang, "Development of a parallelized Adaptive-Mesh Poisson-Boltzmann Equation Solver Using Finite Element Method and Its Applications," Journal of Computational Chemistry, 2006. (SCI, in revision) *.
06. J-S. Wu, **Y.-L. Shao**, P.-C. Huang, T.-C. Cheng, "Development of a Micro-Mixer System and Its Applications," NDL Communication, Vol.12, I1, pp.21-27, 2005. (中文期刊)

Conference Papers:

01. J.-S. Wu and **Y.-L. Shao**, "Assessment of SRT and MRT Scheme in Parallel Lattice Boltzmann Method for Lid-Driven Cavity Flows," The 10th National Computational Fluid Dynamics Conference, Tainan, Taiwan, August, 2002
02. J.-S. Wu and **Y.-L. Shao**, "Comparison of SRT and MRT Schemes in Lattice Boltzmann Method for Simulating the Flow Past a Square in a Channel," The 20th National Conference on Mechanical Engineering The Chinese Society of Mechanical Engineers, NTU, Taipei, Taiwan, December 5-6, 2003
03. J.-S. Wu, T.-C. Cheng, **Y.-L. Shao** and C.-H. Wu, " Development of a Micro-PIV System and Its Applications ", The 11th Symposium on Nano Device Technology (SNDT 2004)
04. J.-S. Wu, **Y.-L. Shao**, C.-H. Wu and T.-C. Cheng, " Design, Manufacture and Performance Analysis for a MicroMixer ", The 11th Symposium on Nano Device Technology (SNDT 2004)
05. J.-S. Wu and **Y.-L. Shao**, "Comparison of the Micro-PIV Measurements with Lattice Boltzmann Method in Micro-Channel Flows," The 11th National CFD

Conference, Tai-Tung, Taiwan, August 5-7, 2004. **(One of the three Best Paper Awards out of 190 papers)**

06. **Y.-L. Shao**, J.-S. Wu and J.-J. Hwang, "The micro-PIV measurement of the passive micro-mixer flow," The Aeromechanics Conference, Kaohsiung, 2004
07. J.-S. Wu, **Y.-L. Shao**, P.-C. Huang and T.-C. Cheng, "A spiral mixer for Microchannels," The 12th Symposium on Nano Device Technology (SNDT 2005)