Parallel computational strategies for modelling transient Stokes fluid flow.

Liu, Wei

CIVIL AND COMPUTATIONAL ENGINEERING
UNIVERSITY OF WALES, SWANSEA

PARALLEL COMPUTATIONAL STRATEGIES FOR MODELLING TRANSIENT STOKES FLUID FLOW

WEI LIU

BSC, COMPUTER SCIENCE AND MATHEMATICS, UNIVERSITY OF LEEDS, LEEDS
MSC, COMPUTATIONAL MODELLING AND FINITE ELEMENT TECHNIQUES IN ENGINEERING MECHANICS, UNIVERSITY OF WALES, SWANSEA

DISSERTATION SUBMITTED TO THE UNIVERSITY OF WALES IN CANDIDATURE FOR THE DEGREE OF DOCTOR OF PHILOSOPHY, 2006

Declaration

This is to certify that this dissertation has not been previously accepted in part or in substance in candidature for any degree and is not concurrently submitted in candidature for any degree at any other university.

Candidate 14/11/06 *Date*

Statement I

This is to certify that except where specific reference to other investigation is made, the work presented in this dissertation is the result of the investigations of the candidate.

Candidate 14/11/06 *Date*

Statement II

I hereby give consent for my dissertation, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Candidate 14/11/06 *Date*

Acknowledgements

I would like to express my sincere gratitude to Prof. D.R.J. Owen and Dr. E.A. de Souza Neto for their guidance, encouragement and support during the course of this research.

I would like to thank all members of the Department of Civil Engineering, University of Wales Swansea for their cooperation in this work. In addition I gratefully acknowledge the support of Rockfield Software Ltd. and their staff for providing facilities and technical assistance during this research work.

Finally, I sincerely thank my father and mother for their understanding and encouragement during my time at university.

To my parents

Synopsis

The present work is centred on two main research areas; the development of finite element techniques for the modelling of transient Stokes flow and implementation of an effective parallel system on distributed memory platforms for solving realistic large-scale Lagrangian flow problems.

The first part of the dissertation presents the space-time Galerkin / least-square finite element implicit formulation for solving incompressible or slightly compressible transient Stokes flow with moving boundaries. The formulation involves a time discontinuous Galerkin method and includes least-square terms in the variational formulation. Since the additional terms involve the residual of the Euler- Lagrangian equations evaluated over element interiors, it prevents numerical oscillation on the pressure field when equal lower order interpolation functions for velocity and pressure fields are used, without violating the Babuška-Brezzi stability condition. The space-time Galerkin / least-square formulation has been successfully extended into the finite element explicit analysis, in which the penalty based discrete element contact algorithm is adopted to simulate fluid-structure or fluid-fluid particle contact.

The second part of the dissertation focuses on the development of an effective parallel processing technique, using the natural algorithm concurrency of finite element formulations. A hybrid iterative direct parallel solver is implemented into the ELFEN/implicit commercial code. The solver is based on a non-overlapping domain decomposition and sub-structure approach. The modified Cholesky factorisation is used to eliminate the unknown variables of the internal nodes at each subdomain and the resulting interfacial equations are solved by a Krylov subspace iterative method. The parallelization of explicit fluid dynamics is based on overlapping domain decomposition and a Schwarz alternating procedure. Due to the dual nature of the overlapping domain decomposition a buffer zone between any two adjacent subdomains is introduced for handling the inter-processor communication. Both solvers are tested on a PC based interconnected network system and its performances are judged by the parallel speed-up and efficiency.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Preliminary Remarks

The finite element method (FEM) is currently one of the most important techniques used for numerical modelling; its application covers almost every aspect within engineering. Together with advances in supercomputer technology FEM has enabled engineers to solve many difficult or previously intractable problems. The popularity of finite element simulation as a way of investigating physical phenomena has been growing steadily. Fluid dynamics research in particular is benefiting from this new methodology. However, the numerical approximation of a physical system, especially replacement of a continuum with a finite number of variables, brings certain approximation errors. At the same time finite element modelling requires an extensive amount of computational time to tackle a realistic large-scale problem with sufficient accuracy. Therefore, a continual effort to develop new computing strategies and techniques for this kind of modelling is still needed in order to achieve substantial speedup and accuracy.

Fluid flow problems that involve changing spatial domains appear in many industrial processes and applications; such as metal forming, glass and polymer forming, mould filling in casting process, liquid sloshing in transportation, food or shampoo container filling etc. Mathematically, it leads to solving an initial boundary value problem for the Navier-Stokes flow in Eulerian coordinates, or the Stokes flow in a Lagrangian frame. In finite element modelling both Eulerian and Lagrangian formulations have

been developed over the years, each having certain limitations. The Eulerian formulation allows large distortion in the fluid motion. Since the Eulerian elements do not deform with the material, no matter how large the deformation is, the elements retain their original shape. This character is most appealing in modelling many manufacturing processes. However, it suffers from two important drawbacks; the convective term that arises due to relative movements between nodal points and material particles, and secondly complex mathematical mappings are required between stationary and moving boundaries. In the Lagrangian formulation the nodes and elements on the finite element mesh move with the material, boundaries and interfaces remain coincident with the element edges, which provides a precise definition of moving boundaries and is also devoid of convective effects. Lagrangian meshes are widely used in solid mechanics, but it is difficult to handle large distortion of a flow domain and a constant remeshing is often required.

A new computational strategy, called space-time Galerkin/least-squares finite element formulation, has emerged in the early nineties and is widely accepted by many researchers and engineers for solving a variety of incompressible flow problems with moving boundaries and interfaces. With error estimator and adaptive remeshing techniques becoming more mature, the difficulties that arise in the Lagrangian formulation can be overcome. The first part of the dissertation is to further improve the performance of the space-time Galerkin/least-squares finite element Lagrangian formulation for a slightly compressible transient Stokes flow, and extend the developed formulation into finite element explicit analysis.

Owing to the extremely intensive computations involved in the flow simulation of realistic large-scale applications, chief among them is the need to solve a large system of linear equations in the implicit analysis. For 3-D fluid flow problems, the number of equations can be easily over hundred of thousands. On the other hand, in the explicit time-integration procedure a time step is much smaller and a very large number (usually over ten million) of time increments have to be imposed for a few seconds of simulation, at a large computational cost. Consequently, the parallel implementation of the solution procedure has become an attractive option for increasing computational capacities, which also becomes feasible due to significant advances in the development of parallel computer hardware, particularly the

emergence of commodity PC clusters. The second part of the dissertation attempts to develop an effective parallel processing technique on distributed memory parallel platforms, utilizing the natural algorithmic concurrency of finite element formulations.

1.2 Scope and Aims

The principal aims of the thesis are centred on three major research areas

1) To extend the space-time Galerkin/least-squares finite element formulation to incompressible or slightly compressible transient Stokes flows including Newtonian or non-Newtonian fluids. The formulation is applied in both implicit and explicit finite element analysis codes.

2) To apply the developed Lagrangian flow formulation and discrete element contact algorithm to simulate large-scale flow problems, which involve constant moving boundaries and contact interfaces, in particular, in fluid-structure and fluid-fluid particle interaction.

3) To develop parallel computational techniques for the solution of large-scale Lagrangian flow problems on a distributed memory platform. The MPI is taken as the message-passing library between processors.

1.3 Thesis Layout

The dissertation consists of six chapters. A brief synopsis of each chapter contained within this dissertation follows.

Chapter 1 outlines the background and the objective of this research; a brief summary of each chapter in the dissertation is included.

Chapter 2 presents the space-time Galerkin/least-squares finite element implicit and explicit formulations for solving incompressible or slightly compressible transient Stokes flow. The basic aspect of the formulation involves a time discontinuous Galerkin method and includes least-squares terms in the variational formulation. The idea is a natural extension of the streamline upwind / Petrov-Galerkin method (SUPG). Since the additional terms involve the residual of the Euler-Lagrangian equations evaluated over element interiors, it prevents numerical oscillation on the pressure field when equal, lower order interpolation functions for velocity and pressure are used. The implicit formulation is highly non-linear due to the prior unknown boundary position, but it can be simplified into a linear, symmetric system of equations by exploiting the algorithmic properties without sacrificing solution accuracy. The extension of the space-time Galerkin/least-squares method into the finite element explicit formulation is also introduced. It provides a powerful tool for simulation of the fluid-structure coupling problems, at the same time avoiding the difficulties raised from contact interaction.

Chapter 3 presents the finite element algorithm of contact modelling for fluids on Lagrangian meshes in the first part. The key aspect in computational contact mechanics is to apply the impenetrability condition to the normal direction of the contact interfaces. In the thesis, the contact modelling is limited to the explicit analysis of the transient Stokes flow problems. The penalty method based discrete element contact algorithm, 2-D or 3-D node-to-facet algorithm, is adopted to simulate fluid-structure or fluid-fluid particle contact. Since the time step is sufficiently small in the explicit time integration procedure, the penalty method is well suited to enforce inequality constraints for the problems that involve large dynamic deformation.

In the second part of the chapter, a continuum adaptive remeshing scheme is presented, which involves definition of a geometry entity related model, error estimate and prediction of mesh density, re-generation of the new mesh by an automatic mesh generator and field values mapping between the old and new meshes. The focus of this part of the chapter is on the introduction of a weighted least squares mapping method, which significantly improves the mapping quality, compared to the background element-mapping scheme.

Chapter 4 describes parallel finite element computational methods for the implicit and explicit solution of the transient Stokes flow problems. Domain decomposition techniques are reviewed. The implementation of an implicit parallel solver, named the hybrid iterative direct parallel solver, is based on a non-overlapping domain decomposition and sub-structure approach; a large scale finite element domain is decomposed into a set of subdomains, the solution of the subdomain problem is naturally parallelized and a direct solver is used. The resulting Schur equations are solved by iterative methods. Several important iterative methods, which are called Krylov subspace projection methods including the Conjugate Gradient (CG), Generalized Minimal Residual (GMRES), Bi-conjugate Gradient (Bi-CG) and Bi-conjugate Gradient Stabilized (Bi-CGSTAB), are introduced in detail.

The parallelization of explicit finite element fluid dynamics with contact conditions is based on the overlapping domain decomposition and the Schwarz alternating procedure. Due to the dual nature of the overlapping partitioning of the domain, communication requirement and computational cost may be slightly more than that of the non-overlapping domain decomposition, but it offers a more efficient and flexible way of dealing with contact problems that appear in fluid-structure and fluid-fluid particles interaction problems.

Chapter 5 illustrates the applicability of the formulations and the algorithms developed with a set of practical examples. The numerical results from finite element analysis are compared with experimental tests. These examples include; horizontal and vertical sloshing water waves in both 2-D and 3-D cases, collapse of a liquid column, 2-D explicit simulation of shampoo filling, etc. The parallel efficiency and scalability on the PC platform are also presented.

Chapter 6 provides an overview of the numerical research performed within this thesis. Some conclusions and suggestions for future developments and improvements are also pointed out.

Chapter 2

Finite Element Formulation For Computational Fluid Dynamics

2.1 Introduction

In recent years, the space-time Galerkin /least-squares finite element formulation for the Navier-Stokes flow with moving boundary problems has been developed by Hughes *et al* [2.1]-[2.4] and Hansbo *et al* [2.5]-[2.6]. The formulation has also been extended for the incompressible Stoke flow by Feng and Peric [2.8], where the convective term is dropped. The formulation is considered as an effective approach to solve a wide class of flow problems involving moving boundaries and interfaces, such as metal forming, glass forming, casting, fluid-structure interactions, fluid particle interaction, free-surfaces and multiple phases [2.9][2.10]. The basic aspects of the formulation uses a time discontinuous Galerkin method and includes least square terms in the variational formulation. Since the shape functions employed are continuous in space but discontinuous in time, the spatial discretization can be changed from one region to another. This feature provides a natural mechanism for incorporating adaptive re-meshing in the formulation. The idea of adding least square terms in the variational formulation is based on the streamline-upwind/Petrov-Galerkin method (SUPG), which was earlier developed by Hughes *et al* [2.11]-[2.13] and Zienkiewicz [2.14][2.15] for convective transport problems. Since the added terms involve residuals of the Euler-Lagrangian equations evaluated over element interiors, it preserves the consistency of the standard Galerkin method. It also prevents numerical oscillation on pressure fields when equal-order interpolation functions for

velocity and pressure are used without enforcing the incompressibility constraints condition. The classical Galerkin variational formulation, which was first proposed by Herrmann[2.16] and may be viewed as a particular case of the Hellinger-Reissner principle [2.17], naturally produced a mixed $\boldsymbol{u}-p$ form. It was recognized that solution of this type of formulation was strongly dependent on the particular pair of velocity and pressure interpolation chosen. In many cases, using equal order interpolation functions for velocity and pressure violates the Babuška-Brezzi stability condition [2.18][2.19]. In particular, this condition rules out the use of lower equal order interpolation, which would be attractive from a computational point of view. The space-time Galerkin /least-squares formulation successfully circumvents the stability conditions and a stable solution can be obtained regardless of the interpolation function employed.

In this chapter the space-time Galerkin /least-squares finite element formulation for a slightly compressible transient Stokes flow is presented. The approach is following the work developed by Feng and Peric [2.8]. The basic unknown variables in the transient Stokes flow are the velocity and pressure, both approximated by C^0 interpolation functions. The final implicit finite element formulation has the same form as described in reference [2.8] except for the addition of an extra pressure mass term, which tends to zero when complete incompressibility is assured. The implicit formulation is highly non-linear due to the prior unknown boundary position and non-Newtonian character of fluid viscosity, which will be discussed in detail in the chapter. The Newton-Raphson approach is adopted to solve the resulting non-linear implicit equations. By exploiting the algorithmic properties, the equations can be degenerated into a linear and symmetric system of equations without sacrificing solution accuracy.

To extend the space-time Galerkin /least-squares method into a finite element explicit formulation is straightforward. In many fluid-structure coupling cases difficulty often arises from contact interaction. Usually, contact forces are calculated within the current spatial configuration. The velocities in the Lagrangian flow are calculated based on the previous configuration, and after solution convergence was reached the current configuration is updated using the obtained velocity vector. This inconsistency

on the different configurations can cause difficulties in obtaining a stable solution in the implicit formulation, but it is not the case for explicit dynamic formulation. Because of its easy treatment of contact conditions, finite element programs based on the explicit dynamic formulation have proved to be a very attractive tool for simulation of fluid-structure interaction problems. The split, characteristic-based scheme for compressible and incompressible flow was proposed by Zienkiewicz et al [2.20]-[2.24], which allows use of a mixed $\boldsymbol{u}-p$ form with lowest equal order interpolations. The key idea of the split operator is to provide a stabilized pressure term for the continuity equation through splitting the momentum balance equations [2.22]. It almost shares the same approximation form with the space-time Galerkin /least-squares method. In this chapter the variational formulation for the space-time Galerkin/least squares method in a space-time domain can be slightly changed and rewritten with an Euler forward integration process. The explicit form of finite element discretization for transient Stokes flow is derived and it can be easily implemented in the finite element explicit code. The stability of the explicit formulation will be also discussed.

An outline of this chapter is arranged as follows; Section 2.2 briefly discusses a slightly compressible form of the Navier-Stokes equations, which represent a strong form of the initial/boundary value problem. Section 2.3 introduces a space-time technique for solving fluid dynamic equations. In section 2.4 the Galerkin/least-squares weighted residual method is employed to form the variational function, which will be later defined in a Lagrangian frame. In section 2.5 an implicit form of the discretized finite element formulation for transient Stoke flow is presented. Section 2.6 examines the kinematics of the moving space-time slabs and linear integration on time. The non-linearity arising from Non-Newtonian flow is discussed in section 2.7. The explicit form of finite element discretization is presented in section 2.8. Finally two and three dimensional equal order elements are presented in section 2.9.

2.2 Fluid dynamics formulation

Consider a viscous, slightly compressible transient fluid with a time domain $t \in [0, \ T]$ and a bounded region $\Omega(t) \in R^{n_{sd}}$ with boundary $\Gamma(t)$, where n_{sd} is the number of spatial dimensions. The Navier- Stokes equations represent the momentum balance and a slightly compressible constraint with the velocity $\boldsymbol{u}(\boldsymbol{x}, \ t)$ and pressure $P(\boldsymbol{x}, \ t)$ as the basic unknown variables, can be written as:

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) - \boldsymbol{f} - \nabla\cdot\boldsymbol{\sigma} = 0 \qquad \text{on } \Omega(t) \ \forall t \in [0, \ T] \qquad 2.1$$

$$\frac{1}{\rho C^2}\frac{\partial P}{\partial t} + \nabla\cdot\boldsymbol{u} = 0 \qquad \text{on } \Omega(t) \ \forall t \in [0, \ T] \qquad 2.2$$

with $$\boldsymbol{u} = u_i \boldsymbol{e}_i \qquad \forall i = 1, n_{sd} \qquad 2.3$$

where $\boldsymbol{e}_i$ denotes an unit vector in Cartesian coordinate direction i, $i = 1, n_{sd}$. ρ is the density of the fluid, $\boldsymbol{\sigma}$ is the Cauchy stress tensor, and $\boldsymbol{f}(\boldsymbol{x}, \ t)$ is the body force per unit mass, $C = \sqrt{K/\rho}$ is the wave speed of the fluid and K is the fluid bulk modulus.

The Dirichlet and Neumann type boundary conditions are defined as

$$\boldsymbol{u} = \boldsymbol{g} \qquad \text{on } \Gamma_g(t) \qquad 2.4$$

$$\boldsymbol{\sigma}\cdot\boldsymbol{n} = \boldsymbol{h} \qquad \text{on } \Gamma_h(t) \qquad 2.5$$

where $\Gamma_g(t)$ and $\Gamma_h(t)$ are complementary subsets of the boundary $\Gamma(t)$ as admitted by the following decomposition

$$\overline{\Gamma_g \cup \Gamma_h} = \Gamma \qquad 2.6$$

and $$\Gamma_g \cap \Gamma_h = 0 \qquad 2.7$$

The unit outward normal vector to $\Gamma(t)$ is denoted by $\boldsymbol{n}=\{n_i\};i=1,n_{sd}$. The initial condition is a divergence-free velocity field specified over the domain $\Omega(t)$ at time $t=0$.

$$\begin{aligned} \boldsymbol{u}(\boldsymbol{x},\ 0)&=\boldsymbol{u}_0 \\ P(\boldsymbol{x},\ 0)&=P_0 \end{aligned} \qquad \text{on } \Omega(0) \qquad 2.8$$

The constitutive relations for fluid are defined by the stress and strain rate tensors as

$$\boldsymbol{\sigma}(\boldsymbol{u},\ P)=2\mu\dot{\boldsymbol{\varepsilon}}(\boldsymbol{u})-P\boldsymbol{I} \qquad 2.9$$

and

$$\dot{\boldsymbol{\varepsilon}}(\boldsymbol{u})=\nabla^s\boldsymbol{u}=\frac{1}{2}\left[\nabla\boldsymbol{u}+(\nabla\boldsymbol{u})^T\right] \qquad 2.10$$

where μ is the viscosity which may be strain rate dependent, and $\boldsymbol{I}$ is an identity tensor. The equations (2.1), (2.2), (2.4), (2.5), (2.8) - (2.10) compose a set of unique and necessary conditions to solve the unknown variables of $\boldsymbol{u}$ and P at time t. For a completely incompressible fluid field equation (2.2) can be simplified as

$$\nabla\cdot\boldsymbol{u}=0 \qquad \text{on } \Omega(t)\ \forall t\in[0,\ T] \qquad 2.11$$

2.3 A space time description of the moving domain

To write the variational form of the space-time formulation for equations (2.1), (2.2) (2.4), (2.5), (2.8)-(2.10), let $I=[0,\ T]$ be an open time interval partitioned into subintervals $I_n=[t_n,\ t_{n+1}]$, where t_n and t_{n+1} belong to an ordered series of time steps $0=t_0<t_1<\cdots<t_N=T$. Let $\Omega(t_n)$ and $\Gamma(t_n)$ be the approximations to the n_{sd} dimension spatial domain Ω with boundary Γ at time t_n. Similarly $\Omega(t_{n+1})$ and $\Gamma(t_{n+1})$ are the approximations at time t_{n+1}, respectively. A space-time slab Q_n is defined by the region enclosed between space domain Ω_n, Ω_{n+1} and lateral boundary P_n, which is the surface described by the boundary $\Gamma(t)$ as t traverses I_n, as shown in Figure 2.1

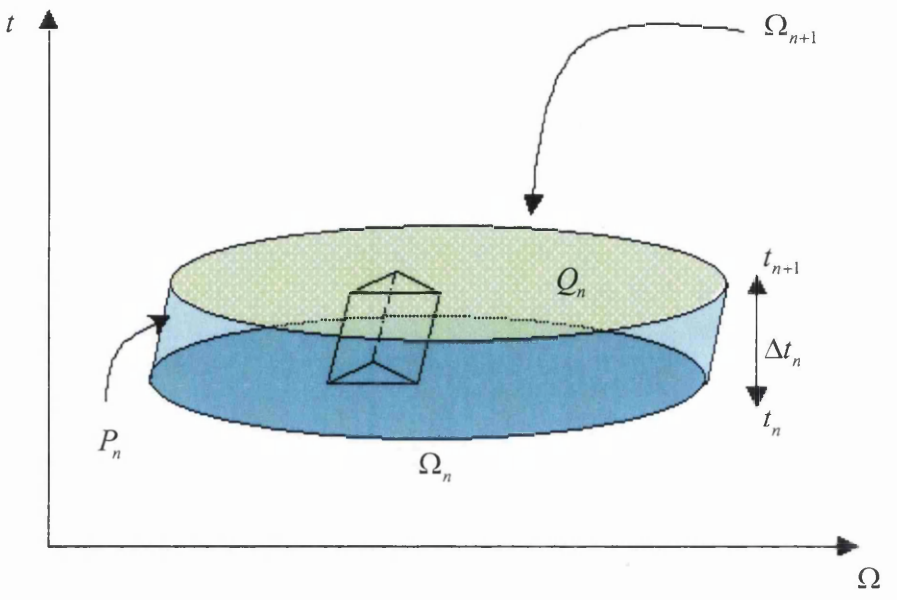


Figure 2.1 A space-time slab Q_n

The lateral boundary P_n is assumed to admit the following decomposition.

$$\overline{P_g \cup P_h} = P_n \tag{2.12}$$

and

$$P_g \cap P_h = 0 \tag{2.13}$$

Let us assume the trial functions $\boldsymbol{u}^h$ and P^h in $(\boldsymbol{x},\ t)$ space are given by,

$$\left(\varphi_{\boldsymbol{u}}^h\right)_n = \left\{ \boldsymbol{u}^h = \left(u_i^h\right) \middle| u_i^h \in H^{lh}(Q_n), u_i^h = g_i^h \text{ on} \left(P_n\right)_{g_i} \ \forall i = 1,\ldots n_{sd} \right\} \tag{2.14}$$

$$\left(\varphi_P^h\right)_n = \left\{ P^h \middle| P^h \in H^{lh}(Q_n) \right\} \tag{2.15}$$

The weighting function $\boldsymbol{w}^h$ and q^h in $(\boldsymbol{x},\ t)$ space are defined by

$$\left(V_{\boldsymbol{u}}^h\right)_n = \left\{ \boldsymbol{w}^h = \left(w_i^h\right) \middle| w_i^h \in H^{lh}(Q_n), w_i^h = 0 \text{ on} \left(P_n\right)_{g_i} \ \forall i = 1,\ldots n_{sd} \right\} \tag{2.16}$$

$$\left(V_P^h\right)_n = \left\{ q^h \middle| q^h \in H^{lh}(Q_n) \right\} \tag{2.17}$$

where $H^{lh}(Q_n)$ represents the admitted finite-dimensional function space over a space-time slab Q_n. At the element domain this space is formed by using first-order

polynomials in space, both trial and weighting functions are continuous in space, but discontinuous in time.

2.4 Variational formulation in the Lagrangian frame

The variational formulation for the space-time Galerkin/least-squares can be written as follows: given $(\boldsymbol{u})_n^-$, $(P)_n^-$ find $\boldsymbol{u}^h \in (\varphi_u^h)_n$ and $P^h \in (\varphi_P^h)_n$ such that $\forall \boldsymbol{w}^h \in (V_u^h)_n$ and $\forall q^h \in (V_P^h)_n$,

$$\begin{aligned}
&\int_{Q_n} \boldsymbol{w}^h \cdot [\rho(\frac{\partial \boldsymbol{u}^h}{\partial t} + \boldsymbol{u}^h \cdot \nabla \boldsymbol{u}^h) - \boldsymbol{f}] dQ + \int_{Q_n} \dot{\boldsymbol{\varepsilon}}(\boldsymbol{w}^h) : \boldsymbol{\sigma}(\boldsymbol{u}^h, P^h) dQ \\
&- \int_{P_n} \boldsymbol{w}^h \cdot \boldsymbol{h} dP + \int_{Q_n} q^h \left(\frac{1}{\rho C^2} \frac{\partial P^h}{\partial t} + \nabla \cdot \boldsymbol{u}^h \right) dQ \\
&+ \sum_{e=1}^{n_e} \int_{Q_n^e} \delta_1 [\rho(\frac{\partial \boldsymbol{w}^h}{\partial t} + \boldsymbol{w}^h \cdot \nabla \boldsymbol{w}^h) - \nabla \cdot \boldsymbol{\sigma}(\boldsymbol{w}^h, q^h)][\rho(\frac{\partial \boldsymbol{u}^h}{\partial t} + \boldsymbol{u}^h \cdot \nabla \boldsymbol{u}^h) - \nabla \cdot \boldsymbol{\sigma}(\boldsymbol{u}_n^h, P_n^h) - \boldsymbol{f}] dQ \\
&+ \sum_{e=1}^{n_e} \int_{Q_n^e} \delta_2 \rho \left(\frac{1}{\rho C^2} \frac{\partial q^h}{\partial t} + \nabla \cdot \boldsymbol{w} \right) \left(\frac{1}{\rho C^2} \frac{\partial P^h}{\partial t} + \nabla \cdot \boldsymbol{u}^h \right) dQ + \int_{\Omega_n} \boldsymbol{w}^h \cdot \rho \left[(\boldsymbol{u}^h)_n^+ - (\boldsymbol{u}^h)_n^- \right] d\Omega \\
&+ \int_{\Omega_n} q^h \frac{1}{\rho C^2} \left[(P^h)_n^+ - (P^h)_n^- \right] d\Omega = 0
\end{aligned} \tag{2.18}$$

where n_e is the number of elements in the domain. δ_1 and δ_2 are non-dimensional stability constants, which will be defined later. The integration process of the equation (2.18) is applied sequentially to all space-time slabs $Q_1, Q_2 \cdots Q_{N-1}$ with

$$(\boldsymbol{u}^h)_n^{\pm} = \lim_{\varepsilon \to 0} \boldsymbol{u}(t_n \pm \varepsilon) \qquad (\boldsymbol{u}^h)_1^- = \boldsymbol{u}_0 \tag{2.19}$$

$$(P^h)_n^{\pm} = \lim_{\varepsilon \to 0} P(t_n \pm \varepsilon) \qquad (P^h)_1^- = P_0 \tag{2.20}$$

The variational form of equation (2.18) can be explained as following,

(a) The first three terms constitute the Galerkin form of the momentum balance equations. The fourth term is the Galerkin form of the continuum equations for a slightly compressible fluid.

(b) The fifth term is the least-squares term of the momentum equations. This term provides stability for the convective dominated case. δ_1 should be $O\left(h^{e2}\right)$ to achieve the best rate of convergence, where h^e is the element characteristic length.

(c) The sixth term is a least-squares term of the continuum equations and it provides stability of the flow formulation with a high Reynolds number [2.25][2.26]. The definition of the stability parameter δ_2 should be a $O\left(h^e\right)$.

(d) The last two terms weakly enforce the continuity of the velocity and pressure field across the space-time slabs.

In the chapter we mainly consider relative slow-speed of viscous fluid that involves moving and deforming spatial configuration with free boundary propagation. The Lagrangian description provides a precise definition of moving boundaries and is devoid of convective effects, due to the mesh moving with the fluid particles. Together with the use of lowest order elements, the variational formulation (2.18) can be simplified as

$$\begin{aligned}
&\int_{Q_n} \dot{\boldsymbol{\varepsilon}}(\boldsymbol{w}^h) : \sigma(\boldsymbol{u}^h, P^h) dQ + \int_{Q_n} q^h \nabla \cdot \boldsymbol{u}^h dQ + \sum_{e=1}^{n_e} \int_{Q_n^e} \delta_1 \nabla q^h \cdot \nabla P^h dQ \\
&+ \sum_{e=1}^{n_e} \int_{Q_n^e} \delta_2 \rho \left(\nabla \cdot \boldsymbol{w}^h\right)\left(\nabla \cdot \boldsymbol{u}^h\right) dQ - \int_{Q_n} (\boldsymbol{w}^h + \delta_1 \nabla q^h) \cdot \boldsymbol{f} dQ - \int_{P_n} \boldsymbol{w}^h \cdot \boldsymbol{h} dP \qquad 2.21 \\
&+ \int_{\Omega_n} \boldsymbol{w}^h \cdot \rho \left[\left(\boldsymbol{u}^h\right)_n^+ - \left(\boldsymbol{u}^h\right)_n^-\right] d\Omega + \int_{\Omega_n} q^h \frac{1}{\rho C^2} \left[\left(P^h\right)_n^+ - \left(P^h\right)_n^-\right] d\Omega = 0
\end{aligned}$$

The derivation of equation (2.21) was based on the following assumptions;

(a) In the Lagrangian frame, the convective terms in equation (2.18) are dropped, leading to

$$\boldsymbol{u}^h \cdot \nabla \boldsymbol{u}^h = \boldsymbol{w}^h \cdot \nabla \boldsymbol{w}^h = 0$$

(b) At each space-time slab, the trial and weighting functions are assumed to be H^1 function and C^0 continuous within space but constant in time, so that their time derivatives will disappear.

$$\frac{\partial \boldsymbol{u}^h}{\partial t} = \frac{\partial P^h}{\partial t} = \frac{\partial \boldsymbol{w}^h}{\partial t} = \frac{\partial q^h}{\partial t} = 0$$

(c) Adoption of linear interpolation functions for 2D triangle and 3D tetrahedral element leads to

$$\nabla \cdot \boldsymbol{\sigma}\left(\boldsymbol{u}^h,\ P^h\right) = -\nabla P$$

2.5 Finite element approximation of the variational formulation

In this section, the finite element equations for transient Stokes flow are developed by means of derivation of the variational function with respective to nodal velocity and pressure respectively. For this purpose the space-time domain is sub-divided into elements Q_n^e, so that the union of the elements comprise the total domain $Q_n = \sum_{e=1}^{N_e} Q_n^e$.

The velocity $\boldsymbol{u}$ and pressure field P is expressed by nodal variables as

$$\boldsymbol{u} = N_b \boldsymbol{I} \boldsymbol{u}^b \qquad u_i = N_b \delta_{il} u_l^b \qquad \forall i,l = 1 \cdots n_{sd} \qquad 2.22$$

$$P = N_b P^b \qquad 2.23$$

where $\boldsymbol{u}^b, P^b$ are the nodal velocity vector and pressure at node b. $N_b = N_b(\boldsymbol{\xi})$ is assumed to be the global nodal shape function temporarily, as shown in Figure 2.2, in order to write the weak form of the variational formulation clearly. Subscripts of the global nodal shape function b range from 1 to n_b, where n_b is the total number of nodes in the domain. Summation over repeated indices is implied. It is emphasized that the global nodal shape function is expressed in terms of reference coordinates in the space-time slab, but constant in time. $\boldsymbol{I}$ is an unit matrix of 2×2 for 2D element and 3×3 for 3D element. The equations (2.22) are given in both tensor notation and indicial notation for clarity. The weight function of virtual velocity and pressure can be also written as,

$$\boldsymbol{w} = N_a \boldsymbol{I} \boldsymbol{w}^a \qquad w_i = N_a \delta_{ik} w_k^a \qquad \forall i,k = 1 \cdots n_{sd} \qquad 2.24$$

$$q = N_a q^a \qquad 2.25$$

where $\boldsymbol{w}^a, q^a$ are the virtual velocity vector and pressure at node a. Subscript of the global nodal shape function $a = 1 \ldots n_a$, $n_a = n_b$.

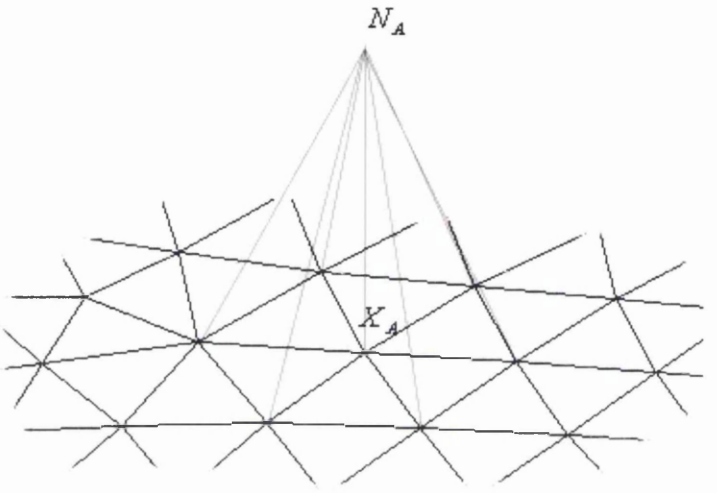


Figure 2.2: Global shape function

The tensor and indicial form of the strain rates and stresses are given by

$$\dot{\boldsymbol{\varepsilon}} = \left(\nabla^s N_b \boldsymbol{I}\right) \boldsymbol{u}^b \qquad \dot{\varepsilon}_{ij} = \frac{1}{2}\left(\frac{\partial N_b}{\partial x_j} u_i^b + \frac{\partial N_b}{\partial x_i} u_j^b\right) \qquad 2.26$$

$$\boldsymbol{\sigma} = 2\mu\left(\nabla^s N_b \boldsymbol{I}\right) \boldsymbol{u}^b - N_b P^b \boldsymbol{I} \quad \sigma_{ij} = \mu\left(\frac{\partial N_b}{\partial x_j} u_i^b + \frac{\partial N_b}{\partial x_i} u_j^b\right) - N_b P^b \delta_{ij} \qquad 2.27$$

Substituting (2.22) –(2.27) into (2.21), the weak form of the variational formulation in Lagrangian description is represented by nodal variables $\boldsymbol{u}^b, P^b, \boldsymbol{w}^a, q^a$ as

$$\varphi\left(w_k^a, q^a\right) = \sum_{a=1}^{n_a} w_k^a \left\{ \int_{Q_n} \mu \sum_{b=1}^{n_b} \left(\frac{\partial N_a}{\partial x_i}\frac{\partial N_b}{\partial x_i}\delta_{kl} + \frac{\partial N_b}{\partial x_k}\frac{\partial N_a}{\partial x_l} \right) u_l^b dQ \right.$$
$$- \int_{Q_n} \sum_{b=1}^{n_b} \left(\frac{\partial N_a}{\partial x_k} N_b \right) P^b dQ + \int_{Q_n} \delta_2 \rho \sum_{b=1}^{n_b} \left(\frac{\partial N_a}{\partial x_k}\frac{\partial N_b}{\partial x_l} \right) u_l^b dQ$$
$$\left. + \int_{\Omega_n} \rho \sum_{b=1}^{n_b} N_a N_b \delta_{kl} \left(u_l^b - u_{l,n-1}^b \right) d\Omega - \int_{Q_n} N_a f_k dQ - \int_{P_n} N_a h_k dP \right\}$$
$$+ \sum_{a=1}^{n_a} q^a \left\{ \int_{Q_n} \sum_{b=1}^{n_b} \left(N_a \frac{\partial N_b}{\partial x_l} \right) u_l^b dQ + \int_{Q_n} \delta_1 \sum_{b=1}^{n_b} \left(\frac{\partial N_a}{\partial x_i}\frac{\partial N_b}{\partial x_i} \right) P^b dQ \right.$$
$$\left. + \int_{\Omega_n} \frac{1}{\rho C^2} \sum_{b=1}^{n_b} N_a N_b \left(P^b - P_{n-1}^b \right) d\Omega - \int_{Q_n} \delta_1 \frac{\partial N_a}{\partial x_i} f_i dQ \right\} \qquad 2.28$$
$$\forall i,k,l = 1 \cdots n_{sd}$$

The variational function $\varphi\left(w_k^a, q^a\right)$ is minimized with respect to the virtual velocity w_k^a, q^a, leading to the expressions

$$\frac{\partial \varphi\left(w_k^a, q^a\right)}{\partial w_k^a} = \int_{Q_n} \mu \sum_{b=1}^{n_b} \left(\frac{\partial N_a}{\partial x_i}\frac{\partial N_b}{\partial x_i}\delta_{kl} + \frac{\partial N_b}{\partial x_k}\frac{\partial N_a}{\partial x_l} \right) u_l^b dQ$$
$$- \int_{Q_n} \sum_{b=1}^{n_b} \left(\frac{\partial N_a}{\partial x_k} N_b \right) P^b dQ + \int_{Q_n} \delta_2 \rho \sum_{b=1}^{n_b} \left(\frac{\partial N_a}{\partial x_k}\frac{\partial N_b}{\partial x_l} \right) u_l^b dQ \qquad 2.29$$
$$+ \int_{\Omega_n} \rho \sum_{b=1}^{n_b} N_a N_b \delta_{kl} \left(u_l^b - u_{l,n-1}^b \right) d\Omega - \int_{Q_n} N_a f_k dQ - \int_{P_n} N_a h_k dP = 0$$
$$\forall \quad k = 1, \ldots n_{sd}; \qquad a = 1, 2 \ldots n_a$$

$$\frac{\partial \varphi\left(w_k^a, q^a\right)}{\partial q^a} = \int_{Q_n} \sum_{b=1}^{n_b} \left(N_a \frac{\partial N_b}{\partial x_l} \right) u_l^b dQ + \int_{Q_n} \delta_1 \sum_{b=1}^{n_b} \left(\frac{\partial N_a}{\partial x_i}\frac{\partial N_b}{\partial x_i} \right) P^b dQ$$
$$+ \int_{\Omega_n} \frac{1}{\rho C^2} \sum_{b=1}^{n_b} N_a N_b \left(P^b - P_{n-1}^b \right) d\Omega - \int_{Q_n} \delta_1 \frac{\partial N_a}{\partial x_i} f_i dQ = 0 \qquad 2.30$$
$$\forall \quad a = 1, 2, \ldots \; n_a$$

Equation (2.29) is a Galerkin/least-squares discretization of the momentum balance equation in the k-th direction of node a. There are $n_{sd} \times n_a$ equations of this form. Equation (2.30) is a pressure equation for node a and there are n_a equations of this form. The discretized implicit equations for transient Stokes flow will be defined as a mixed form

$$\begin{bmatrix} \boldsymbol{M}+\boldsymbol{K} & \boldsymbol{Q} \\ \boldsymbol{Q}^T & -(\boldsymbol{M}_P+\boldsymbol{M}_\rho) \end{bmatrix} \begin{bmatrix} \overline{\boldsymbol{U}} \\ \overline{\boldsymbol{P}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_u \\ \boldsymbol{F}_P \end{bmatrix} \tag{2.31}$$

where $\overline{\boldsymbol{U}}$ and $\overline{\boldsymbol{P}}$ are respectively the vector of unknown nodal values of velocity $\boldsymbol{u}$ and pressure P; $\boldsymbol{M}$ is the well-known mass matrix. The stiffness matrix $\boldsymbol{K}$ is composed of a linear viscosity matrix and a velocity stabilization matrix, which is derived from the least-squares term of the continuity equation. For Non-Newtonian fluid, the stiffness matrix $\boldsymbol{K}$ also includes a nonlinear viscosity matrix and will be discussed in section 2.7. $\boldsymbol{Q}$ is a coupling matrix relating nodal velocity with nodal pressure. $\boldsymbol{M}_P$ is a pressure stabilization matrix and $\boldsymbol{M}_\rho$ is a similar mass matrix corresponding to nodal pressure.($\boldsymbol{M}_\rho$ becomes zero when complete incompressibility is assured.) These terms can be assembled by elemental contribution to the appropriate location in the global matrices as

$$\begin{aligned} \boldsymbol{K}_{ab} &= \boldsymbol{K}_{ab}^{lin} + \boldsymbol{K}_{ab}^{s} \\ &= \int_{Q_n} u[(\nabla N_a \cdot \nabla N_b)\boldsymbol{I} + (\nabla N_b \otimes \nabla N_a)]dQ + \int_{Q_n} \delta_2 \rho (\nabla N_a \otimes \nabla N_b) dQ \end{aligned} \tag{2.32a}$$

$$\boldsymbol{M}_{ab} = \int_{\Omega_n} \rho N_a N_b \boldsymbol{I} d\Omega \tag{2.32b}$$

$$\boldsymbol{Q}_{ab} = -\int_{Q_n} \nabla N_a N_b dQ \tag{2.32c}$$

$$\boldsymbol{M}_{P,ab} = \int_{Q_n} \delta_1 \nabla N_a \cdot \nabla N_b dQ \tag{2.32d}$$

$$\boldsymbol{M}_{\rho,ab} = \int_{\Omega_n} \frac{1}{\rho C^2} N_a N_b d\Omega \tag{2.32e}$$

$$\boldsymbol{F}_{u,a} = \int_{Q_n} N_a \boldsymbol{f} dQ_n + \int_{P_n} N_a \boldsymbol{h} dP + \int_{\Omega_n} \sum_{b=1}^{n_b} \rho N_a N_b \boldsymbol{u}_{n-1}^b d\Omega \tag{2.32f}$$

$$F_{P,a} = -\int_{Q_n} \delta_1 \nabla N_a \cdot \boldsymbol{f} dQ - \int_{\Omega_n} \frac{1}{\rho C^2} \sum_{b=1}^{n_b} N_a N_b P_{n-1}^b d\Omega \tag{2.32g}$$

where subscript *ab* stands for a cross term of stiffness or mass matrix for nodes *a* and *b*. The subscript of *a* represents the corresponding velocity and pressure force vector at node *a*. If the pressure mass matrix $\boldsymbol{M}_{\rho,ab}$ and second term of the right hand side of

equation (2.32g) are dropped, equations (2.31) are the same as the one given by reference [2.8]. Normally, those terms for the Stokes flow could be dropped in the implicit formulation, but it is essential for the explicit dynamic formulation. Following the works of Hansbo [2.25], the parameters δ_1 and δ_2 are set to be

$$\delta_1 = \alpha h_e^2, \quad \delta_2 = h_e \tag{2.33}$$

where h_e is the characteristic length of element, α is a problem dependent constant.

2.6 Reference space-time domain and linear integration in time

In equation (2.31) and (2.32), all the integrations are conducted over an unknown evolving domain Q_n, which is defined by the region enclosed by space domain Ω_n and unknown domain Ω_{n+1} as described in section 2.2. In order to solve this problem, several authors [2.1][2.2][2.8] introduce a reference space-time domain $\overline{Q}_n \in R^{n_{sd}} \times [0, \ T]$ and a mapping function $\phi_n : \overline{Q}_n \to Q_n$ from the reference space-time element in the $(\chi, \ t)$ coordinate system on to the deformed physical element in the $(x, \ t)$ coordinate system, i.e. the motion of $\overline{Q}_n$ is defined by

$$(x, \ t) = \phi_n(\chi, \ \tau) \tag{2.34}$$

where the mapping function $\phi_n(\chi, \ \tau)$ is taken as

$$\phi_n(\chi, \ \tau) = (\chi + \tau u_n, \ t - t_n) \tag{2.35}$$

where u_n is the velocity field to be determined in the current space-time slab n. χ is the reference coordinates, normally it is taken as $\chi = x_n$, the initial spatial domain of the space-time slab Q_n. Therefore $\overline{Q}_n = \Omega_n \times [0, \ \Delta t_n]$, where $\Delta t_n = t_{n+1} - t_n$. For the reference space-time domain, the time is an additional dimension, the deformation gradients is a $(n_{sd}+1)\times(n_{sd}+1)$ matrix of partial derivatives of the mapping function ϕ_n. Assuming f_n is a space-time deformation gradient matrix and F_n is a spatial

deformation gradient matrix, the space-time deformation gradient matrix f_n at time $\tau \in [0, \quad \Delta t]$ can be defined as

$$f_n = \begin{bmatrix} \frac{\partial x}{\partial \chi} & \frac{\partial x}{\partial \tau} \\ \frac{\partial t}{\partial \chi} & \frac{\partial t}{\partial \tau} \end{bmatrix} = \begin{bmatrix} F_n & u_n \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.36}$$

where

$$F_n(\chi, \quad \tau) = \frac{\partial x}{\partial \chi} = I + \tau \frac{\partial u_n(\chi, \tau)}{\partial \chi} \tag{2.37}$$

The Jacobian determinants of the deformation gradient f_n and F_n are denoted by $\overline{J}_n(\chi, \quad \tau)$ and $J_n(\chi, \quad \tau)$ respectively, as

$$\overline{J}_n(\chi, \quad \tau) = \det f_n(\chi, \quad \tau) = \det F_n(\chi, \quad \tau) = J_n(\chi, \quad \tau) \tag{2.38}$$

Then all the integrations in equation (2.32) can be conducted over the reference space-time domain $\overline{Q}_n$ instead of the unknown evolving space-time domain Q_n. Within an element space-time domain, the integration can be written as

$$\begin{aligned} \int_{Q_n^e} (\cdot) dQ &= \int_{\overline{Q}_n^e} (\cdot) J_n(\chi, \quad \tau) d\overline{Q} \\ &= \int_{\Delta t} \int_{\Omega_n^e} (\cdot) J_n(\chi, \quad \tau) d\Omega d\tau \end{aligned} \tag{2.39}$$

Since we chose a lower order interpolation function as the element nodal shape function, equation (2.39) can be further rewritten as

$$\begin{aligned} \int_{\Delta t} \int_{\Omega_n^e} (\cdot) J_n(\chi, \quad \tau) d\Omega d\tau &= \int_{\Delta t} J_n(\tau) \int_{\Omega_n^e} (\cdot) d\Omega d\tau \\ &= \beta_e \int_{\Omega_n^e} (\cdot) d\Omega \end{aligned} \tag{2.40a}$$

where

$$\beta_e = \int_{\Delta t} J_n(\tau) d\tau \tag{2.40b}$$

This means that the element Jacobian determinant $J_n = J_n(\tau)$ is no longer dependent on the spatial position χ and is only a linear function of time τ. It can be proved as

follows. In the element spatial domain Ω_n^e the velocity vector $\boldsymbol{u}^e$ is expressed by element nodal variables as

$$\boldsymbol{u}^e = N_{n_d} \boldsymbol{I} \boldsymbol{u}^{n_d} \qquad \boldsymbol{u}_i^e = N_{n_d} \delta_{ij} \boldsymbol{u}_j^{n_d} \quad \forall i, j = 1, n_{sd} \tag{2.41}$$

where $n_d = 1,3$ for a lower order triangle element. According to equation (2.37), the spatial deformation gradient is defined as

$$F_{ij} = \delta_{ij} + \tau \frac{\partial u_i^e}{\partial \chi_j} = \delta_{ij} + \tau \frac{\partial N_{n_d}}{\partial \chi_i} u_j^{n_d} \tag{2.42}$$

Since N_{n_d} is a linear function in terms of χ_i and its derivative is constant, $\boldsymbol{F}_n$ is constant with respect to χ and only linearly dependent of the time τ, so as is its determinant $\boldsymbol{J}_n(\tau)$. The function β_e for each element could be linearly integrated in time.

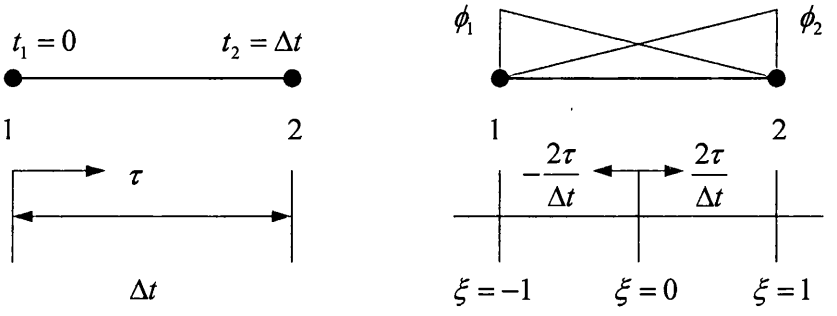


Figure 2.3 One dimensional time element

Considering a Δt interval, the time τ can be taken by a linear interpolation, as shown in Figure 2.3.

$$\tau = \phi_1(\xi) t_1 + \phi_2(\xi) t_2 = \phi_2(\xi) \Delta t \tag{2.43}$$

and

$$\phi_1 = \frac{1}{2}(1-\xi) \quad \phi_2 = \frac{1}{2}(1+\xi) \tag{2.44}$$

where $\phi_i, i = 1,2$ is a linear shape function related with local coordinate ξ. The integration of the element function β_e is carried out on the local coordinate system as

$$\beta_e = \int_0^{\Delta t} \boldsymbol{J}_n(\tau)\, d\tau = \int_{-1}^{1} \boldsymbol{J}_n(\xi) \frac{\partial \tau}{\partial \xi} d\xi$$
$$= \frac{\Delta t}{2} \int_{-1}^{1} \boldsymbol{J}_n(\xi)\, d\xi = \frac{\Delta t}{2} \sum_{k=1}^{2} w_k \boldsymbol{J}_n(\xi_k) \qquad 2.45$$

and

$$\boldsymbol{J}_n(\xi_k) = \det\left[\delta_{ij} + \phi_2(\xi_k) \Delta t \frac{\partial N_{nd}}{\partial \chi_i} u_j^{nd} \right] \qquad 2.46$$
$$\forall i, j = 1, n_{sd}$$

where w_k is a weighting function and ξ_k is a local coordinate at Gauss point *k*.

Remark:

(1) From Equation (2.42) it is clearly shown that the Jacobian determinant of $\boldsymbol{J}_n(\tau)$ is dependent on unknown variables, i.e. nodal velocity $\boldsymbol{u}$, therefore the mixed $\boldsymbol{u}-p$ form of equations (2.31) is a non-linear system of equations. The Newton-Raphson scheme is a natural choice for solving such non-linear implicit equations. In the reference [2.8], linearization of $\boldsymbol{J}_n(\tau)$ leads to an un-symmetric stiffness and an un-symmetric load stiffness matrix in the formulation, which should be solved by an un-symmetric equation solver within the space-time slab.

(2) The Jacobian determinant of $\boldsymbol{J}_n(\tau)$ measures the ratio of the spatial volume $V^e(\tau)$ at time τ with reference volume $V_0^e(\tau)$ at $\tau = 0$, i.e. $J^e(\tau) = \frac{V^e(\tau)}{V_0^e}$. For a completely incompressible fluid $\boldsymbol{J}(\tau)$ must be equal to 1, or $\boldsymbol{J}_n(\tau)$ would be very close to 1 for a slightly incompressible medium. From our experience we set $\boldsymbol{J}_n(\tau) = 1$ without severely sacrificing solution accuracy. If the viscosity of the fluid is constant, equation (2.31) will be linear and symmetric, which is equivalent to solving a linear Stokes flow problem. It can be solved using a symmetric equation solver within the space-time slab.

(3) In an explicit dynamic code, the critical time step Δt is normally controlled by the element characteristic length and the wave speed of the medium, it is

relatively smaller than the one used in the implicit code. The natural setting is $J_n(\tau)=1$ and $\beta_e=\Delta t$ for incompressible or slightly incompressible fluid in the explicit dynamic code. Equation (2.31) would be simply transformed to a forward Euler integration procedure, which will be discussed in detail at section 2.8.

2.7 Non-Newtonian fluid

The viscosity for non-Newtonian fluids can be represented in the following general form as

$$\mu=\mu(\dot{r}) \tag{2.47}$$

where the viscosity μ is generally dependent on the shear rate $\dot{r}$, which is set by

$$\dot{r}=\left(2\dot{\varepsilon}_{ij}\dot{\varepsilon}_{ij}\right)^{\frac{1}{2}} \tag{2.48}$$

There are a large number of mathematical models that have been developed for modelling varies type of non-Newtonian fluid [2.26]. The Power law, Bingham fluid model and Herschel-Bulkley fluid models represent the most well known examples.

Power Law Fluid Model

The power law is one of the most widely used non-Newtonian fluid models. The viscosity and the strain rate are fitted as a linear relationship on a logarithm scale. It can be written as

$$\begin{aligned}&\mu(\dot{r})=\mathbb{C}\dot{r}^{n-1} \qquad \dot{r}>\dot{r}_c\\ &\mu(\dot{r})=\mathbb{C}\left[\dot{r}-\dot{r}_c\right]\left[(n-1)\dot{r}_c^{(n-2)}\right]\dot{r}_c^{n-1}+\mathbb{C}\dot{r}^{(n-1)} \qquad \dot{r}<\dot{r}_c\end{aligned} \tag{2.49}$$

where $\mathbb{C}$ represents the fluid consistency index and n is a power law index. (For most shearing fluids $n<1$.) $\dot{r}_c$ is a critical shear rate. The equation (2.49) has a limitation, since it only fits the experimental data within a special region of interest. The regions

that will not be defined by the power law model are those at shear rate $\dot{r} = 0$ and $\dot{r} \to \infty$.

Bingham Fluid Model

The Bingham fluid model is one of the visco-plastic type of fluid models. The most widely used Bingham model is provided by Papanastasiou [2.27], it can be represented by the following equation

$$\mu(\dot{r}) = \mu_0 + \frac{\sigma_Y}{\dot{r}}(1 - e^{-m\dot{r}}) \qquad 2.50$$

where μ_0 is the initial viscosity, m is a stress growth exponent, while σ_Y is the yielding stress. These parameters can be fitted empirically. As the exponent m becomes larger, as shown in Figure 2.4, the viscosity μ expresses a visco-plastic type behaviour.

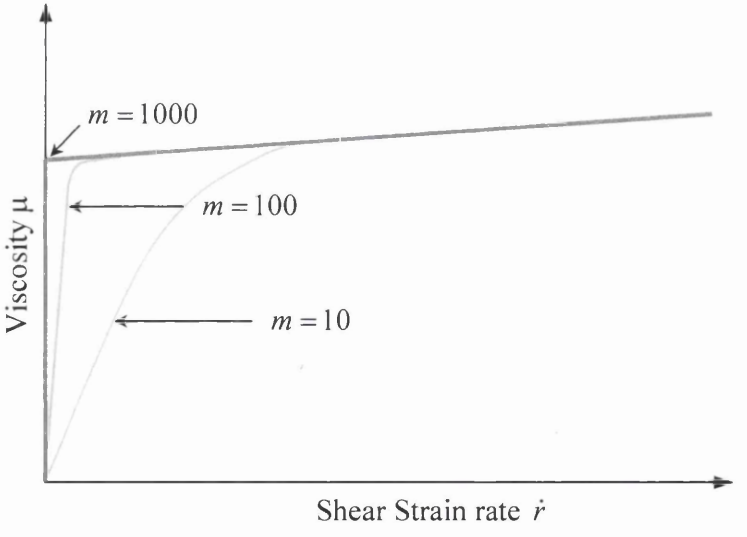


Figure 2.4 Bingham fluid model

Herschel-Bulkley Fluid Model

The Herschel-Bulkley fluid model is one of the shear thinning fluid models, which combines the power law and the Bingham model. Chhabra [2.28] describes it with the following form

$$\mu(\dot{r}) = \frac{\sigma_Y}{\dot{r}}(1 - e^{-m\dot{r}}) + \mathbb{C}\dot{r}^{n-1} \tag{2.51}$$

where the parameters σ_Y, $\mathbb{C}$, m and n have the same meaning as the Bingham and Power law models.

It has already been shown that there is a high non-linearity in the viscous part of governing equation (2.29), as a consequence of the non-Newtonian character of the viscosity $\mu(\dot{r})$. The successful linearization of this term is essential to obtain quadratic rate of asymptotic convergence on the Newton-Raphson iterations. It is also important in accurately modelling non-Newtonian fluid behaviour. The deviatoric part of the first term in equation (2.21) can be re-written as

$$\dot{\varepsilon}(\boldsymbol{w}^h) : 2\mu\dot{\boldsymbol{\varepsilon}}(\boldsymbol{u}^h, P^h) = \nabla \boldsymbol{w}^h : 2\mu\dot{\boldsymbol{\varepsilon}}(\boldsymbol{u}^h) \tag{2.52}$$

Differential of both side of equation (2.52) gives

$$\delta\left[\nabla \boldsymbol{w}^h : 2\mu\dot{\boldsymbol{\varepsilon}}(\boldsymbol{u}^h)\right] = \nabla \boldsymbol{w}^h : 2\mu\delta\dot{\boldsymbol{\varepsilon}} + \frac{4\mu'}{\dot{r}}\left[\nabla \boldsymbol{w} : \dot{\boldsymbol{\varepsilon}}(\boldsymbol{u}^h)\right]\left[\dot{\boldsymbol{\varepsilon}} : \nabla \delta \boldsymbol{u}\right] \tag{2.53}$$

Substituting equations (2.22) and (2.24) into (2.53), the first term in the right hand side of equation (2.53) is derived as

$$\begin{aligned} \nabla \boldsymbol{w}^h : 2\mu\delta\dot{\boldsymbol{\varepsilon}}\left(\boldsymbol{u}^h\right) &= \mu\left(\frac{\partial w_i}{\partial x_j}\frac{\partial \delta u_i}{\partial x_j} + \frac{\partial w_i}{\partial x_j}\frac{\partial \delta u_j}{\partial x_i}\right) \\ &= \mu\left(w_k^a \frac{\partial N_a}{\partial x_j}\frac{\partial N_b}{\partial x_j}\delta_{kl}\delta u_l^b + w_k^a \frac{\partial N_a}{\partial x_j}\frac{\partial N_b}{\partial x_i}\delta_{ik}\delta_{jl}\delta u_l^b\right) \\ &= w_k^a \mu\left[\left(\nabla N_a \cdot \nabla N_b\right)\boldsymbol{I} + \left(\nabla N_a \otimes \nabla N_b\right)\right]\delta \boldsymbol{u}^b \end{aligned} \tag{2.54}$$

The second term in the right hand side of equation (2.53) is derived as

$$\begin{aligned} \frac{4\mu'}{\dot{r}}\left[\nabla \boldsymbol{w}^h : \dot{\boldsymbol{\varepsilon}}\left(\boldsymbol{u}^h\right)\right]\left[\dot{\boldsymbol{\varepsilon}} : \nabla \delta \boldsymbol{u}^h\right] &= \frac{4\mu'}{\dot{r}}\left(w_k^a \frac{\partial N_a}{\partial x_j}\dot{\varepsilon}_{kj}\right)\left(\dot{\varepsilon}_{lj}\frac{\partial N_b}{\partial x_j}\delta u_l^b\right) \\ &= w_k^a\left[\frac{4\mu'}{\dot{r}}\left(\dot{\boldsymbol{\varepsilon}} \cdot \nabla N_a\right) \otimes \left(\dot{\boldsymbol{\varepsilon}} \cdot \nabla N_b\right)\right]\delta \boldsymbol{u}^b \end{aligned} \tag{2.55}$$

The stiffness matrix given by (2.54) is the viscous parts of the stiffness matrix described in equation (2.32a), an additional part of the nonlinear viscosity stiffness matrix due to linearization of fluid viscosity is given by equation (2.55) as

$$K_{ab}^{non} = \int_{Q_n} \frac{4\mu'}{\dot{r}} (\dot{\boldsymbol{\varepsilon}} \cdot \nabla N_a) \otimes (\dot{\boldsymbol{\varepsilon}} \cdot \nabla N_b) dQ \qquad 2.56$$

where the derivative of the viscosity $\mu' \left(= \frac{\partial u}{\partial \dot{r}} \right)$ with respective to the shear rate is dependent on the non-Newtonian fluid model. The following equations illustrate the value for μ' with different type of non-Newtonian fluids

- ***Power Law Fluid***

$$\mu'(\dot{r}) = \mathbb{C}(n-1)\dot{r}^{(n-2)} \qquad \dot{r} \geq \dot{r}_c$$

$$\mu'(\dot{r}) = \mathbb{C}\left[(n-1)\dot{r}_c^{(n-2)}\right]\dot{r}_c^{n-1} + \mathbb{C}(n-1)\dot{r}^{(n-2)} \qquad \dot{r} < \dot{r}_c \qquad 2.57$$

- ***Bingham Fluid***

$$\mu'(\dot{r}) = \frac{\sigma_Y}{\dot{r}}\left[e^{-m\dot{r}}\left(\frac{1}{\dot{r}} + m\right) - \frac{1}{\dot{r}}\right] \qquad 2.58$$

- ***Herschel-Bulkley Fluid***

$$\mu'(\dot{r}) = \frac{\sigma_Y}{\dot{r}}\left[e^{-m\dot{r}}\left(\frac{1}{\dot{r}} + m\right) - \frac{1}{\dot{r}}\right] + \mathbb{C}(n-1)\dot{r}^{(n-2)} \qquad 2.59$$

The equation (2.56) will be added to (2.32a) to give a consistent stiffness matrix for the non-Newtonian fluids. The equations (2.31) can be solved using a Newton Raphson type iterative scheme. A typical algorithm for solving equations within a space–time slab Q_n is summarized in Table 2.1.

Loop over all slabs: For $n = 1,2,\ldots,$ until $t_{n+1} = T$ DO

i. Set iteration count $i = 0$

ii. Evaluate residual forces using equations

$$\boldsymbol{\psi}_u^{(i)} = \boldsymbol{F}_u - \left(\boldsymbol{M} + \boldsymbol{K} + \boldsymbol{K}^{non}\right)\boldsymbol{U}_n^{(i)} - \boldsymbol{Q}\boldsymbol{P}_n^{(i)}$$

$$\boldsymbol{\psi}_P^{(i)} = \boldsymbol{F}_P - \boldsymbol{Q}^T\boldsymbol{U}_n^{(i)} + \left(\boldsymbol{M}_P + \boldsymbol{M}_\rho\right)\boldsymbol{P}_n^{(i)}$$

when $i = 0$, $\quad \boldsymbol{U}_n^{(0)} = \boldsymbol{U}_{n-1} \quad \boldsymbol{P}_n^{(0)} = \boldsymbol{P}_{n-1}$

iii. Using iterative solver to solve

$$\begin{bmatrix} \boldsymbol{M} + \boldsymbol{K} + \boldsymbol{K}^{non} & \boldsymbol{Q} \\ \boldsymbol{Q}^T & -\left(\boldsymbol{M}_P + \boldsymbol{M}_\rho\right) \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{U}_n^{(i)} \\ \delta\boldsymbol{P}_n^{(i)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\psi}_u^{(i)} \\ \boldsymbol{\psi}_P^{(i)} \end{bmatrix}$$

iv. Update velocity and pressure vectors

$$\boldsymbol{U}_n^{(i+1)} = \boldsymbol{U}_n^{(i)} + \delta\boldsymbol{U}_n$$

$$\boldsymbol{P}_n^{(i+1)} = \boldsymbol{P}_n^{(i)} + \delta\boldsymbol{P}_n$$

v. If $\delta\boldsymbol{U}^{(i)}, \delta\boldsymbol{P}^{(i)}$ or $\boldsymbol{\psi}_u^{(i)}, \boldsymbol{\psi}_P^{(i)}$ do not satisfy the convergence condition, then set $i = i + 1$ and go to step (ii), otherwise update coordinates.

vi. $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \Delta t \cdot \boldsymbol{U}_n^{(i+1)}$

EndDo

Table 2.1 Newton-Raphson iterative algorithm

If a constant viscosity is used for Newtonian fluids, i.e. $\mu' = 0$, then K^{non} will disappear in Table 2.1.

2.8 Explicit discretization of fluid dynamics formulation

The mixed $\boldsymbol{u} - P$ form of fluid dynamic formulation is presented in equation (2.31), which has almost the same form with the one defined in reference [2.8]. It is an implicit non-linear system of equations and should be solved iteratively. When considering the case of fluid contact with other bodies, such as fluid-structure interaction, fluid-particles interaction, the problems will be extremely difficult to solve implicitly due to contact forces. It is necessary to establish an explicit form of

fluid dynamic formulation. The variational formulation (2.21) for the space-time Galerkin/least squares in slab Q_n can be slightly changed and rewritten as

$$\begin{aligned}&\int_{Q_n}\dot{\boldsymbol{\varepsilon}}(\boldsymbol{w}^h):\sigma(\boldsymbol{u}^h,P^h)dQ+\int_{Q_n}q^h\nabla\cdot\boldsymbol{u}^h dQ+\sum_{e=1}^{n_e}\int_{Q_n^e}\delta_1\nabla q^h\cdot\nabla P^h dQ\\&+\sum_{e=1}^{n_e}\int_{Q_n^e}\delta_2\rho\left(\nabla\cdot\boldsymbol{w}^h\right)\left(\nabla\cdot\boldsymbol{u}^h\right)dQ-\int_{Q_n}(\boldsymbol{w}^h+\delta_1\nabla q^h)\cdot\boldsymbol{f}dQ-\int_{P_n}\boldsymbol{w}^h\cdot\boldsymbol{h}dP\\&+\int_{\Omega_n}\boldsymbol{w}^h\cdot\rho\left[\left(\boldsymbol{u}^h\right)_{n+1}^{+}-\left(\boldsymbol{u}^h\right)_{n+1}^{-}\right]d\Omega+\int_{\Omega_n}q^h\frac{1}{\rho C^2}\left[\left(P^h\right)_{n+1}^{+}-\left(P^h\right)_{n+1}^{-}\right]d\Omega=0\end{aligned}\qquad 2.60$$

where the Euler forward integration process is adopted for the last two terms in equation (2.60) compared with (2.21). It is satisfied sequentially to all space-time slabs $Q_1,Q_2\cdots Q_n$ with

$$\left(\boldsymbol{u}^h\right)_{n+1}^{\pm}=\lim_{\varepsilon\to 0}\boldsymbol{u}\left(t_{n+1}\pm\varepsilon\right)\qquad\left(\boldsymbol{u}^h\right)_1^{-}=\boldsymbol{u}_0\qquad 2.61$$

$$\left(P^h\right)_{n+1}^{\pm}=\lim_{\varepsilon\to 0}P\left(t_{n+1}\pm\varepsilon\right)\qquad\left(P^h\right)_1^{-}=P_0\qquad 2.62$$

The derivation is based on the assumption of the element Jacobian determinant $\boldsymbol{J}_n(\tau)=1$ and the parameter $\beta_e=\Delta t$, therefore the equation (2.39) and (2.40a) can be further simplified as

$$\int_{Q_n^e}(\bullet)dQ=\int_{\Delta t}\boldsymbol{J}_n(\tau)\int_{\Omega_n^e}(\bullet)d\Omega d\tau=\Delta t\int_{\Omega_n^e}(\bullet)d\Omega\qquad 2.63$$

Following the derivation described in section 2.4 the explicit discretization of the fluid dynamic formulation is given by

$$\begin{bmatrix}\boldsymbol{M}&0\\0&-\boldsymbol{M}_\rho\end{bmatrix}\begin{bmatrix}\dfrac{\bar{\boldsymbol{U}}_{n+1}-\bar{\boldsymbol{U}}_n}{\Delta t}\\[2ex]\dfrac{\bar{\boldsymbol{P}}_{n+1}-\bar{\boldsymbol{P}}_n}{\Delta t}\end{bmatrix}+\begin{bmatrix}{}^*\boldsymbol{K}&{}^*\boldsymbol{Q}\\{}^*\boldsymbol{Q}^T&-\boldsymbol{M}_P\end{bmatrix}\begin{bmatrix}\bar{\boldsymbol{U}}_n\\\bar{\boldsymbol{P}}_n\end{bmatrix}=\begin{bmatrix}{}^*\boldsymbol{F}_u\\{}^*\boldsymbol{F}_P\end{bmatrix}\qquad 2.64$$

or

$$\begin{aligned}\bar{\boldsymbol{U}}_{n+1}&=\bar{\boldsymbol{U}}_n+\Delta t\boldsymbol{M}^{-1}\left[{}^*\boldsymbol{F}_u-\left({}^*\boldsymbol{K}\bar{\boldsymbol{U}}_n+{}^*\boldsymbol{Q}\bar{\boldsymbol{P}}_n\right)\right]\\\bar{\boldsymbol{P}}_{n+1}&=\bar{\boldsymbol{P}}_n+\Delta t\boldsymbol{M}_\rho^{-1}\left[-{}^*\boldsymbol{F}_P-\left(-{}^*\boldsymbol{Q}^T\bar{\boldsymbol{U}}_n+{}^*\boldsymbol{M}_P\bar{\boldsymbol{P}}_n\right)\right]\end{aligned}\qquad 2.65$$

where $\bar{\boldsymbol{U}}_{n+1}$ is an unknown nodal velocity vector with dimension $n_{sd} \times n_a$, $\bar{\boldsymbol{P}}_{n+1}$ is an unknown nodal pressure vector with dimension n_a at time $n+1$. $\bar{\boldsymbol{U}}_n$ and $\bar{\boldsymbol{P}}_n$ are the nodal velocity vector and pressure vector, respectively, at time n. The mass matrices $\boldsymbol{M}$ and $\boldsymbol{M}_\rho$ are already defined by equation (2.32b) and (2.32e). The stiffness matrix ${}^*\boldsymbol{K}$, pressure stabilized matrix ${}^*\boldsymbol{M}_P$, velocity-pressure coupling matrix ${}^*\boldsymbol{Q}$ and load vectors ${}^*\boldsymbol{F}_u$ and ${}^*\boldsymbol{F}_P$, which are integrated over the reference spatial domain Ω_n only, are defined as follows,

$$\begin{aligned} {}^*\boldsymbol{K}_{ab} &= {}^*\boldsymbol{K}_{ab}^{lin} + {}^*\boldsymbol{K}_{ab}^{s} \\ &= \int_{\Omega_n} u[(\nabla N_a \cdot \nabla N_b)\boldsymbol{I} + (\nabla N_b \otimes \nabla N_a)]d\Omega + \int_{\Omega_n} \delta_2 \rho (\nabla N_a \otimes \nabla N_b) d\Omega \end{aligned} \quad 2.66a$$

$${}^*\boldsymbol{Q}_{ab} = -\int_{\Omega_n} \nabla N_a N_b d\Omega \quad 2.66b$$

$${}^*\boldsymbol{M}_{P,ab} = \int_{\Omega_n} \delta_1 \nabla N_a \cdot \nabla N_b d\Omega \quad 2.66c$$

$${}^*\boldsymbol{F}_{u,a} = \int_{\Omega_n} N_a \boldsymbol{f} d\Omega + \int_{\Gamma_n} N_a \boldsymbol{h} d\Gamma \quad 2.66d$$

$${}^*F_{P,a} = -\int_{\Omega_n} \delta_1 \nabla N_a \cdot \boldsymbol{f} d\Omega \quad 2.66e$$

where subscript *ab* represents a cross term of stiffness or mass matrix for nodes *a* and *b*. The subscript *a* represents the corresponding velocity and pressure force vector for node *a*. The differences between the equation (2.32) and (2.66) are that equations (2.32) are integrated over a space–time domain Q_n and the equations (2.66) are integrated over a reference spatial domain Ω_n. Two pivoting mass matrices in equation (2.64) or (2.65) are generally diagonalized to permit a completely explicit solution.

Remark:

(1) With a fully incompressible fluid we note that pressure mass matrix $\boldsymbol{M}_\rho$ in equation (2.64) tends zero, which is not allowed in the explicit time

integration procedure, i.e. the pressure mass matrix $\boldsymbol{M}_p$ is necessary for the explicit fluid dynamic analysis.

(2) If a standard Galerkin finite element approximation is applied to the Stokes flow, we have the well known mixed $\boldsymbol{u}-P$ form [2.16][2.24] with $\boldsymbol{M}_P = 0$ in equation (2.64) or (2.65). This restricts using mixed interpolation or requires special integration procedure in the case when the pressure variable is eliminated at the element level by a penalty method [2.29]. The difficulty will be removed by introducing a stabilized non–zero matrix $\boldsymbol{M}_P$.

(3) The formulation presented in the chapter generalized the Petrov-Galerkin method developed by Hughes *et al* [2.11]-[2.13], which circumvents the Babuška -Brezzi condition in the context of Stoke flows and guarantees a stable solution for using simple equal order interpolation functions for both velocity and pressure variables [2.1]-[2.8].

The forward Euler integration in equation (2.65) can be easily implemented and is very robust, by which we mean that the explicit procedure seldom aborts due to failure of the numerical algorithm. The price you pay for this simplicity is the conditional stability of the explicit method. If the time step Δt exceeds a critical time step Δt_{crit}, solution will quickly diverge.

A stable time step for a mesh with constant strain elements is given by

$$\Delta t = \eta \Delta t_{crit} \qquad \Delta t_{crit} = \min({}^1\Delta t_{crit}, {}^2\Delta t_{crit}) \tag{2.67}$$

and

$${}^1\Delta t_{crit} = \frac{2}{\omega_{\max}} \le \min_e \frac{h_e}{C_e} \tag{2.68}$$

$${}^2\Delta t_{crit} = \min_e \frac{2h_e}{3\alpha K} \tag{2.69}$$

where ω_{max} is the maximum frequency of the linearized system, h_e is a characteristic length of the element, C_e is the current wave speed in the element and η is a reduction factor that accounts for the destabilizing effects of nonlinearities, a good choice for η is $0.7 \le \eta \le 0.9$. α is a problem dependent constant, which is defined in equation (2.33). Equation (2.68) means that the time necessary for the sound speed wave to traverse the element and equation (2.69) is derived from a transient diffusion problem. In the case of slightly compressible fluids such as water for instance, over ten million time steps were needed to simulate a water tank sloshing over 8.0 second – at a large computational expense.

2.9 Two and three dimensional equal order u-P mixed elements

In section 2.5 and 2.8 the finite element formulation of transient Stokes flow for both implicit and explicit analysis were described. Now the details of how the problems (2.31) and (2.65) are solved will be illustrated with a two-dimensional C^0 triangle and three-dimensional C^0 tetrahedral iso-parametric elements.

2.9.1 3-nodal constant strain triangle element

The linear, constant strain triangular element is based on a standard linear polynomial for approximation of both velocity vector $\boldsymbol{u} = \{u, \quad v\}$ and pressure P in the element. The element shape functions are defined by a set of local coordinates system as;

$$\begin{aligned} N_1(\boldsymbol{\xi}) &= N_1(\xi,\eta) = 1-\xi-\eta \\ N_2(\boldsymbol{\xi}) &= N_2(\xi,\eta) = \xi \\ N_3(\boldsymbol{\xi}) &= N_3(\xi,\eta) = \eta \end{aligned} \qquad 2.70$$

The element geometry is defined by its nodal coordinates $\boldsymbol{x}^a$ and its corresponding shape functions, as shown in Figure 2.5.

$$x(\xi) = \sum_{a=1}^{3} N_a(\xi) x^a \tag{2.71}$$

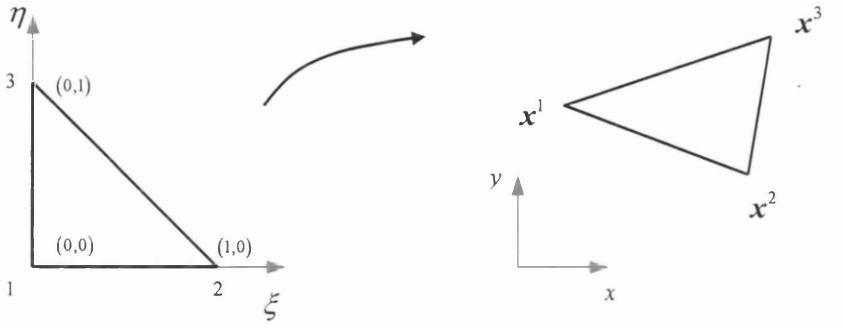


Figure 2.5 A 3-nodal, constant strain triangle element

In the element the velocity and pressure fields are also expressed in terms of local coordinate system ξ, not Cartesian coordinate system x. The derivatives of the shape function with respect to x in equations (2.32) and (2.66) can be defined by the *chain rule* as

$$\nabla N_a = \begin{bmatrix} \frac{\partial N_a}{\partial x} \\ \frac{\partial N_a}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial N_a}{\partial \xi} \\ \frac{\partial N_a}{\partial \eta} \end{bmatrix} \tag{2.72}$$

where the first term in the right side of equation (2.72) is an inverse matrix of the Jacobian matrix, which is calculated from the element Jacobian matrix as,

$$\boldsymbol{J}^{(e)} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \sum_{a=1}^{3} \frac{\partial N_a}{\partial \xi} x^a & \sum_{a=1}^{3} \frac{\partial N_a}{\partial \xi} y^a \\ \sum_{a=1}^{3} \frac{\partial N_a}{\partial \eta} x^a & \sum_{a=1}^{3} \frac{\partial N_a}{\partial \eta} y^a \end{bmatrix} = \begin{bmatrix} x_{21} & y_{21} \\ x_{31} & y_{31} \end{bmatrix} \tag{2.73}$$

Then the inverse of $\boldsymbol{J}^{(e)}$ can be written as

$$\left[\boldsymbol{J}^{(e)}\right]^{-1} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{bmatrix} = \frac{1}{2A^{(e)}} \begin{bmatrix} y_{31} & -y_{21} \\ -x_{31} & x_{21} \end{bmatrix} \tag{2.74}$$

Since

$$2A^{(e)} = x_{21}y_{31} - x_{31}y_{21}$$

where $A^{(e)}$ is the area of the triangular element, now the derivatives of the shape function with respect to Cartesian coordinates x can be defined explicitly, as

$$\nabla N_1 = \frac{1}{2A^{(e)}}\begin{bmatrix} y_{23} \\ x_{32} \end{bmatrix} \quad \nabla N_2 = \frac{1}{2A^{(e)}}\begin{bmatrix} y_{31} \\ x_{13} \end{bmatrix} \quad \nabla N_3 = \frac{1}{2A^{(e)}}\begin{bmatrix} y_{12} \\ x_{21} \end{bmatrix} \tag{2.75}$$

where

$$x_{ab} = x^a - x^b \qquad y_{ab} = y^a - y^b \tag{2.76}$$

2.9.2 4-nodal constant strain tetrahedral element

The 4-nodal tetrahedral element is a constant strain, 3D element, its shape functions are defined by a local coordinate system ξ as

$$\begin{aligned} N_1(\xi) &= N_1(\xi,\eta,\zeta) = 1-\xi-\eta-\zeta \\ N_2(\xi) &= N_2(\xi,\eta,\zeta) = \xi \\ N_3(\xi) &= N_3(\xi,\eta,\zeta) = \eta \\ N_4(\xi) &= N_4(\xi,\eta,\zeta) = \zeta \end{aligned} \tag{2.77}$$

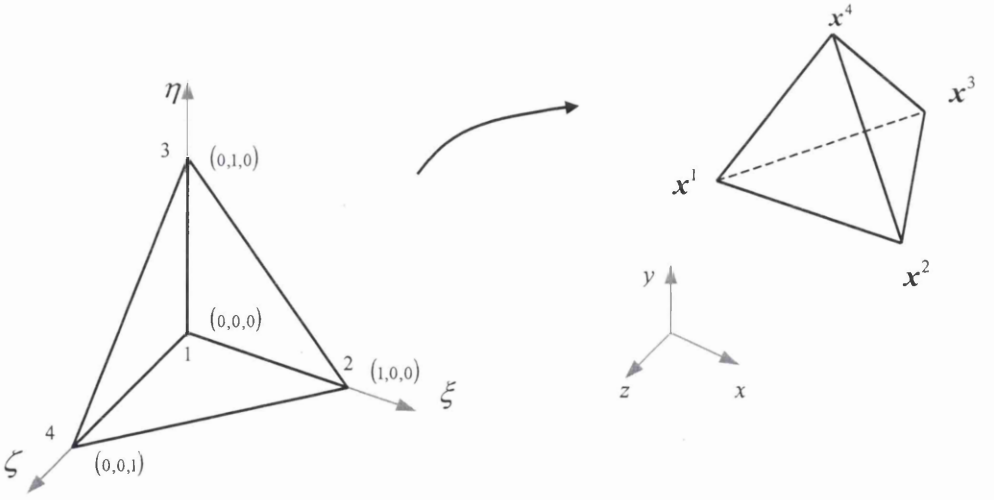


Figure 2.6 A 4-nodal, constant strain tetrahedral element

The element geometry is defined by its nodal coordinates $\boldsymbol{x}^a$ same as the 2-D triangle element, Figure 2.6 shows element Cartesian coordinate mapping from the local coordinate system.

$$\boldsymbol{x}(\xi) = \sum_{a=1}^{4} N_a(\xi)\boldsymbol{x}^a \tag{2.78}$$

The Jacobian matrix for the tetrahedral element can be written as

$$\boldsymbol{J}^{(e)} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} = \begin{bmatrix} \sum_{a=1}^{4} \frac{\partial N_a}{\partial \xi} x^a & \sum_{a=1}^{4} \frac{\partial N_a}{\partial \xi} y^a & \sum_{a=1}^{4} \frac{\partial N_a}{\partial \xi} z^a \\ \sum_{a=1}^{4} \frac{\partial N_a}{\partial \eta} x^a & \sum_{a=1}^{4} \frac{\partial N_a}{\partial \eta} y^a & \sum_{a=1}^{4} \frac{\partial N_a}{\partial \eta} z^a \\ \sum_{a=1}^{4} \frac{\partial N_a}{\partial \zeta} x^a & \sum_{a=1}^{4} \frac{\partial N_a}{\partial \zeta} y^a & \sum_{a=1}^{4} \frac{\partial N_a}{\partial \zeta} z^a \end{bmatrix} = \begin{bmatrix} x_{21} & y_{21} & z_{21} \\ x_{31} & y_{31} & z_{31} \\ x_{41} & y_{41} & z_{41} \end{bmatrix} \tag{2.79}$$

The inverse matrix $\left[\boldsymbol{J}^{(e)}\right]^{-1}$ can be derived explicitly as

$$\left[\boldsymbol{J}^{(e)}\right]^{-1} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} & \frac{\partial \zeta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} & \frac{\partial \zeta}{\partial y} \\ \frac{\partial \xi}{\partial z} & \frac{\partial \eta}{\partial z} & \frac{\partial \zeta}{\partial z} \end{bmatrix} = \frac{1}{6V^{(e)}} \begin{bmatrix} D_{11} & -D_{21} & D_{31} \\ -D_{12} & D_{22} & -D_{32} \\ D_{13} & -D_{23} & D_{33} \end{bmatrix} \tag{2.80}$$

$$6V = x_{21}y_{31}z_{41} + y_{21}z_{31}x_{41} + x_{31}y_{41}z_{21} - (x_{41}y_{31}z_{21} + y_{21}z_{41}x_{31} + x_{21}y_{41}z_{31}) \tag{2.81}$$

where $V^{(e)}$ is an element volume. D_{ki} are the minors of a_{ki} in $\boldsymbol{J}^{(e)}$, they are defined as follows;

$$
\begin{aligned}
D_{11} &= y_{31}z_{41} - y_{41}z_{31} \\
D_{21} &= y_{21}z_{41} - y_{41}z_{21} \\
D_{31} &= y_{31}z_{21} - y_{21}z_{31} \\
D_{12} &= x_{31}z_{41} - x_{41}z_{31} \\
D_{22} &= x_{21}z_{41} - x_{41}z_{21} \\
D_{32} &= x_{21}z_{31} - x_{31}z_{21} \\
D_{13} &= x_{31}y_{41} - x_{41}y_{31} \\
D_{23} &= x_{21}y_{41} - x_{41}y_{21} \\
D_{33} &= x_{21}y_{31} - x_{31}y_{21}
\end{aligned}
\qquad 2.82
$$

Now the derivatives of the shape function with respect to Cartesian coordinates $\boldsymbol{x}$ can be defined explicitly, for the 4-noded tetrahedral element, as

$$
\nabla N_2 = \frac{1}{6V^{(e)}}\begin{bmatrix} D_{11} \\ -D_{12} \\ D_{13} \end{bmatrix} \quad \nabla N_3 = \frac{1}{6V^{(e)}}\begin{bmatrix} -D_{21} \\ D_{22} \\ -D_{23} \end{bmatrix} \quad \nabla N_4 = \frac{1}{6V^{(e)}}\begin{bmatrix} D_{31} \\ -D_{32} \\ D_{33} \end{bmatrix} \qquad 2.83
$$

According to equation (2.77) the gradient of shape function N_1 is given by

$$
\nabla N_1 = -\nabla N_2 - \nabla N_3 - \nabla N_4 \qquad 2.84
$$

2.9.3 Mass matrix for velocity and pressure terms

The element mass matrix is evaluated in the configuration Ω_n at time t_n for the element space-time slab $Q_n^{(e)}$. The consistent mass matrices for velocity and pressure terms are given by equation (2.32b) and (2.32e), respectively, as,

$$
\boldsymbol{M}^e = \left\{\boldsymbol{M}_{ab}^e\right\} = \left\{\int_{\Omega_n^e} \rho N_a N_b \boldsymbol{I} d\Omega\right\} = \int_{\Omega_n^e} \rho \boldsymbol{N}_u^T \boldsymbol{N}_u d\Omega \qquad 2.85
$$

$$
\boldsymbol{M}_\rho^e = \left\{M_{\rho,ab}^e\right\} = \left\{\int_{\Omega_n^e} \frac{1}{\rho C^2} N_a N_b d\Omega\right\} = \int_{\Omega_n^e} \frac{1}{\rho C^2} \boldsymbol{N}_p^T \boldsymbol{N}_p d\Omega \qquad 2.86
$$

and

$$\boldsymbol{N}_u = \left[N_1 \boldsymbol{I}, N_2 \boldsymbol{I}, \cdots, N_{n_d} \boldsymbol{I} \right] \tag{2.87a}$$

$$\boldsymbol{N}_p = \left[N_1, N_2, \cdots, N_{n_d} \right] \tag{2.87b}$$

where N_a and N_b are the shape function at nodes a and b, they are already defined in the previous sections. $\boldsymbol{N}_u$ and $\boldsymbol{N}_p$ are the shape function matrices for the velocity and pressure term, respectively, $\boldsymbol{I}$ is the unit matrix with dimension 2×2 for the 2D element and 3×3 for the 3D element. $\boldsymbol{M}^e_{ab}$ is the $n_{sd}\times n_{sd}$ sub-matrix and $M^e_{\rho,ab}$ is a 1×1 sub-matrix, subscript n_{sd} denotes the number of degrees of freedom per node for the velocity term and n_d denotes the number of nodes per element. In equation (2.85) the symbol $\{\ \}$ represents assembling of sub-matrices, the operator will be applied to equations in later sub-sections. The component of the consistent mass sub-matrices $\boldsymbol{M}^e_{ab}$ and $M^e_{\rho,ab}$ can be calculated explicitly for a triangular or tetrahedral element by using the formulae given by references [2.15],

2d case
$$\int_{\Omega_e} L_1^m L_2^n L_3^p d\Omega = 2A^{(e)} d \frac{m!n!p!}{(m+n+p+2)!} \tag{2.88}$$

3d case
$$\int_{\Omega_e} L_1^m L_2^n L_3^p L_4^q d\Omega = 6V^{(e)} \frac{m!n!p!q!}{(m+n+p+q+3)!} \tag{2.89}$$

where L_i is the nodal shape function at node i. Also d denotes the thickness in the two dimensional case, it is equal to 1 for plane strain problem and to $2\pi x_c$ or $2\pi y_c$ for axisymmetric problems, which is dependent on axisymmetric about the x or y axis. The centre point coordinates x_c, y_c in the triangular element are linearly interpolated by the element nodal coordinates. When the integration formulae (2.88) or (2.89) is applied to (2.85), the component of $\boldsymbol{M}^e_{ab}$ can be written as,

2d case
$$M^e_{ab} = \begin{cases} 2A^{(e)}\rho \dfrac{2!}{(2+2)!} = \dfrac{1}{6} A^{(e)} \rho d & a = b \\ 2A^{(e)}\rho \dfrac{1!1!}{(1+1+2)!} = \dfrac{1}{12} A^{(e)} \rho d & a \neq b \end{cases} \tag{2.90}$$

3d case
$$M^e_{ab} = \begin{cases} 6V^{(e)}\rho\dfrac{2!}{(2+3)!} = \dfrac{1}{10}V^{(e)}\rho & a = b \\ 6V^{(e)}\rho\dfrac{1!1!}{(1+1+3)!} = \dfrac{1}{20}V^{(e)}\rho & a \neq b \end{cases} \qquad 2.91$$

The calculation of the component of $M_{\rho,ab}$ follows the same procedure as M^e_{ab}. The diagonal or lumped mass matrices for velocity and pressure terms M^e_{ab} and $M^e_{\rho,ab}$ can be obtained by the row-sum technique [2.30], giving

2d case
$$M^e_{aa} = \frac{1}{3}A^{(e)}\rho d \qquad M^e_{\rho,aa} = \frac{1}{3\rho C^2}A^{(e)}d \qquad 2.92$$

3d case
$$M^e_{aa} = \frac{1}{4}V^{(e)}\rho \qquad M^e_{\rho,aa} = \frac{1}{4\rho C^2}V^{(e)} \qquad 2.93$$

The lumped mass matrices defined in equation (2.92) and (2.93) will be used in the explicit fluid formulation (2.65).

2.9.4 Velocity-pressure coupling matrix

In the discrete momentum and continuum equations, the element velocity-pressure coupling matrix $\boldsymbol{Q}^e$ or $(\boldsymbol{Q}^e)^T$ is calculated within the element space-time slab $Q^{(e)}_n$, it was given by (2.32c) as

$$\begin{aligned} \boldsymbol{Q}^e = \left\{\boldsymbol{Q}^e_{ab}\right\} &= -\left\{\int_{Q_n} \nabla N_a N_b dQ\right\} \\ &= -\beta^e\left\{\int_{\Omega_n} \nabla N_a N_b d\Omega\right\} = \beta^e \int_{\Omega_n} \boldsymbol{BmN}_p d\Omega \end{aligned} \qquad 2.94$$

where the element function β_e is already defined by equation (2.45). $\boldsymbol{Q}^e_{ab}$ is the $n_{sd}\times 1$ sub-matrix. The element velocity-pressure coupling matrix $\boldsymbol{Q}^e$ is a $n_{evb}\times n_d$ matrix, where n_{evb} is equal to the product of n_{sd} and n_d. The strain-velocity matrix $\boldsymbol{B}$ and

transformation vector $\boldsymbol{m}$ for 2D plane strain, axisymmetric and 3D element can be defined as

2d plane strain and axisymmetric element

$$\boldsymbol{B}=[\boldsymbol{B}_1,\boldsymbol{B}_2,\boldsymbol{B}_3];\quad \boldsymbol{B}_a=\begin{bmatrix} \frac{\partial N_a}{\partial x} & 0 \\ 0 & \frac{\partial N_a}{\partial y} \\ \frac{\partial N_a}{\partial y} & \frac{\partial N_a}{\partial x} \\ \frac{N_a}{x_c} & 0 \end{bmatrix} \quad a=1,3 \qquad 2.95a$$

$$\boldsymbol{m}=\begin{bmatrix}1, & 1, & 0, & 1\end{bmatrix}^T \qquad 2.95b$$

3d solid element

$$\boldsymbol{B}=[\boldsymbol{B}_1,\boldsymbol{B}_2,\boldsymbol{B}_3,\boldsymbol{B}_4];\quad \boldsymbol{B}_a=\begin{bmatrix} \frac{\partial N_a}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_a}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_a}{\partial z} \\ \frac{\partial N_a}{\partial y} & \frac{\partial N_a}{\partial x} & 0 \\ 0 & \frac{\partial N_a}{\partial z} & \frac{\partial N_a}{\partial y} \\ \frac{\partial N_a}{\partial z} & 0 & \frac{\partial N_a}{\partial x} \end{bmatrix} \quad a=1,4 \qquad 2.96a$$

$$\boldsymbol{m}=\begin{bmatrix}1, & 1, & 1, & 0, & 0, & 0\end{bmatrix}^T \qquad 2.96b$$

2.9.5 Linear and non-linear stiffness matrix

In the discrete momentum equations, the element combined stiffness matrix $\boldsymbol{K}^e$ is calculated within the element space-time slab $Q_n^{(e)}$, it was given by (2.32a) and (2.56) as,

$$
\begin{aligned}
\boldsymbol{K}^e &= \boldsymbol{K}^{e,lin} + \boldsymbol{K}^{e,s} + \boldsymbol{K}^{e,non} = \left\{ \boldsymbol{K}_{ab}^{e,lin} + \boldsymbol{K}_{ab}^{e,s} + \boldsymbol{K}_{ab}^{e,non} \right\} \\
&= \left\{ \int_{Q_n^e} u[(\nabla N_a \cdot \nabla N_b)\boldsymbol{I} + (\nabla N_b \otimes \nabla N_a)]dQ + \int_{Q_n^e} \delta_2 \rho (\nabla N_a \otimes \nabla N_b)\, dQ + \right. \\
&\quad \left. \int_{Q_n^e} \frac{4\mu'}{\dot{r}} (\dot{\boldsymbol{\varepsilon}} \cdot \nabla N_a) \otimes (\dot{\boldsymbol{\varepsilon}} \cdot \nabla N_b) dQ \right\} \\
&= \beta^e \left\{ \int_{\Omega_n^e} u[(\nabla N_a \cdot \nabla N_b)\boldsymbol{I} + (\nabla N_b \otimes \nabla N_a)]d\Omega + \int_{\Omega_n^e} \delta_2 \rho (\nabla N_a \otimes \nabla N_b)\, d\Omega + \right. \\
&\quad \left. \int_{\Omega_n^e} \frac{4\mu'}{\dot{r}} (\dot{\boldsymbol{\varepsilon}} \cdot \nabla N_a) \otimes (\dot{\boldsymbol{\varepsilon}} \cdot \nabla N_b) d\Omega \right\} \\
&= \beta^e \int_{\Omega_n^e} \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B} d\Omega + \int_{\Omega_n^e} \delta_2 \rho \left(\boldsymbol{B}^T \boldsymbol{m} \right) \left(\boldsymbol{m}^T \boldsymbol{B} \right) d\Omega + \\
&\quad \int_{\Omega_n^e} \frac{4\mu'}{\dot{r}} \left(\boldsymbol{B}^T \hat{\boldsymbol{\varepsilon}} \right) \left(\hat{\boldsymbol{\varepsilon}}^{\mathrm{T}} \boldsymbol{B} \right) d\Omega
\end{aligned}
\qquad 2.97
$$

where the viscosity matrix $\boldsymbol{D}$ and strain rate vector $\hat{\boldsymbol{\varepsilon}}$ of 2D plane strain, axisymmetric and 3D solid elements are given by

2d plane strain and axisymmetric element

$$
\boldsymbol{D} = \begin{bmatrix} 2\mu & 0 & 0 & 0 \\ 0 & 2\mu & 0 & 0 \\ 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 2\mu \end{bmatrix} \qquad 2.98a
$$

$$
\hat{\boldsymbol{\varepsilon}}^T = \left[\dot{\varepsilon}_{11},\ \dot{\varepsilon}_{22},\ \dot{\varepsilon}_{12},\ \dot{\varepsilon}_{33} \right] \qquad 2.98b
$$

3d solid element

$$
\boldsymbol{D} = \begin{bmatrix} 2\mu & 0 & 0 & 0 & 0 & 0 \\ 0 & 2\mu & 0 & 0 & 0 & 0 \\ 0 & 0 & 2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \qquad 2.99a
$$

$$\hat{\boldsymbol{\varepsilon}}^T = \left[\dot{\varepsilon}_{11},\ \dot{\varepsilon}_{22},\ \dot{\varepsilon}_{33},\ \dot{\varepsilon}_{12},\ \dot{\varepsilon}_{23},\ \dot{\varepsilon}_{13}\right] \qquad 2.99b$$

where $\boldsymbol{K}^{e,lin}$, $\boldsymbol{K}^{e,s}$ and $\boldsymbol{K}^{e,non}$ are the element linear viscosity, velocity stabilization and non-linear viscosity stiffness matrices, respectively, with dimension $n_{evb} \times n_{evb}$, they are also all symmetric matrices. $\boldsymbol{K}_{ab}^{e,lin}$, $\boldsymbol{K}_{ab}^{e,s}$ and $\boldsymbol{K}_{ab}^{e,non}$ are the element stiffness sub-matrices with dimension $n_{sd} \times n_{sd}$,

2.9.6 Pressure stabilization matrix

In the continuum equations, the element pressure stabilization matrix $\boldsymbol{M}_p^e$ is evaluated within the element space-time slab $Q_n^{(e)}$, it was given by (2.32d) as

$$\begin{aligned} \boldsymbol{M}_p^e = \left\{M_{p,ab}^e\right\} &= \left\{ \int_{Q_n^e} \delta_1 \nabla N_a \cdot \nabla N_b dQ \right\} \\ &= \beta^e \left\{ \int_{\Omega_n^e} \delta_1 \nabla N_a \cdot \nabla N_b d\Omega \right\} = \beta^e \int_{\Omega_n^e} \delta_1 \left(\nabla \boldsymbol{N}_p\right)^T \cdot \left(\nabla \boldsymbol{N}_p\right) d\Omega \end{aligned} \qquad 2.100$$

where the size of the element pressure stabilization matrix $\boldsymbol{M}_p^e$ is $n_d \times n_d$. $M_{p,ab}^e$ is a 1×1 sub-matrix. The spatial gradient operator for 2D and 3D cases are defined as,

$$\nabla^T = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right] \quad or \quad \nabla^T = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right] \qquad 2.101$$

2.10 References

[2.1] T.J.R. Hughes, L.P. Franca and G.M. Hulbert, A new finite element formulation for computational fluid dynamics: VIII. The Galerkin/least-squares method for advective-diffusive equations, *Comput. Methods. Appl. Mech. Engrg.* 73 (1989) 173-189.

[2.2] T.J.R. Hughes, L.P. Franca and M. Mallet, A new finite element formulation for computational fluid dynamics: I. Symmetric forms of the compressible Euler and Navier-Stokes equations and the second law of thermodynamics, *Comput. Methods. Appl. Mech. Engrg.* 54 (1986) 223-234.

[2.3] T.J.R. Hughes and G.M. Hulbert, Space-time finite element methods for elastodynamics: formulations and error estimates, *Comput. Methods. Appl. Mech. Engrg.* 66 (1988) 339-363.

[2.4] Arif Masud and T.J.R. Hughes, A space-time Galerkin/least-squares finite element formulation of the Navier-Stokes equations for moving domain problems, *Comput. Methods. Appl. Mech. Engrg.* 146 (1997) 91-126.

[2.5] P. Hansbo, The characteristic streamline diffusion method for the time-dependent incompressible Navier-Stokes equations, *Comput. Methods. Appl. Mech. Engrg.* 99 (1992) 171-186.

[2.6] P. Hansbo and A. Szepessy, A velocity-pressure streamline diffusion method for the incompressible Navier-Stokes equations, *Comput. Methods. Appl. Mech. Engrg.* 84 (1990) 175-192.

[2.7] T.E. Tezduyar, J. Liou and M. Behr, A new strategy for finite element computations involving moving boundaries and interfaces – the DSD/ST procedure: II. Computation of free surface flows, two-liquid flows, and flows with drifting cylinders, *Comput. Methods. Appl. Mech. Engrg.* 94 (1992) 353-371.

[2.8] Y.T. Feng and D. Peric, A time-adaptive space-time finite element method for incompressible Lagrangian flows with free surfaces: Computational issues, *Comput. Methods. Appl. Mech. Engrg.* 190 (2000) 499-518.

[2.9] A.A. Johnson and T.E. Tezduyar, Simulation of multiple spheres falling in a liquid-filled tube, *Comput. Methods. Appl. Mech. Engrg.* 134 (1996) 351-373.

[2.10] T.E. Tezduyar, J. Liou and M. Behr, A new strategy for finite element computations involving moving boundaries and interfaces – the DSD/ST procedure: I. The concept and the preliminary test, *Comput. Methods. Appl. Mech. Engrg.* 94 (1992) 339-351.

[2.11] T.J.R. Hughes and A.N. Brooks, A theoretical framework for Petrov-Galerkin method for discontinuous weighting functions: application to the streamline upwind procedure, in: R.H. Gallagher, D.H. Norrie, J.T. Oden and O.C. Zienkiewicz, eds., *Finite Elements in Fluids*, Vol. IV (Wiley, London, 1985) 46-65.

[2.12] T.J.R. Hughes, L.P. Franca and M. Balestra, A new finite element formulation for computational fluid dynamics: V. Circumventing the Babuska-Brezzi Condition: A Stable Petrov-Galerkin Formulation of the Stokes Problem Accommodating Equal-Order Interpolations. *Comput. Methods. Appl. Mech. Engrg.* 59 (1986) 85-99.

[2.13] C. Johnson, Streamline diffusion methods for problems in fluid mechanics, in: R.H. Gallagher, G. Carey, J.T. Oden and O.C. Zienkiewicz, eds., *Finite Elements in Fluids*, Vol. VI (Wiley, London, 1985) 251-261.

[2.14] O.C. Zienkiewicz, The Finite Element Method. 3rd Edition (McGraw-Hill, London, 1977).

[2.15] O.C. Zienkiewicz and R.L. Taylor, The Finite Element Method. (Butterworth-Heinemann, Oxford, 2000).

[2.16] L.R. Herrmann, Elasticity equations for nearly incompressible elasticity, AIAA J. 3 (1965) 1896-1900.

[2.17] E. Hellinger, Der Allgemeinen Ansätze der Mechanik der Kontinua, *Encyclopädie der Mathematischen Wissenschaften*, Vol.4, Part 4 (Teubner, Leipzig, 1914) 602-694.

[2.18] I. Babuška, The finite element method with Lagrangian multipliers, *Numerische Mathematik*, 20 (1973) 179-192.

[2.19] F. Brezzi and K.J. Bathe, A discourse on the stability conditions for mixed finite element formulations, *Comput. Methods. Appl. Mech. Engrg.* 82 (1990) 27-57.

[2.20] O.C. Zienkiewicz and R. Codina, A general algorithm for compressible and incompressible flow – Part 1 the split characteristics-based scheme, *Int. J. Numer. Fluids*. 20 (1995) 869-885.

[2.21] J. Rojek, O.C. Zienkiewicz, E. Onate and E. Postek, Advances in FE explicit formulation for simulation of metalforming processes, *J. Materials Processing Technology*. 119 (2001) 41-47.

[2.22] O.C. Zienkiewicz, J. Rojek, R.L. Taylor and M. Pastor, Triangles and tetrahedral in explicit dynamic codes for solids, *Comput. Methods. Appl. Mech. Engrg*. 43 (1998) 565-583.

[2.23] R. Codina, M. Vasquez and O.C. Zienkiewicz, A fractional step method for compressible flow: Boundary conditions and incompressible limit, *Proc. Int. Conf. On Finite Elements in Fluids – New Trends and Applications*, Venezia (1995) 409-418.

[2.24] O.C. Zienkiewicz, K. Morgan, B.V.K. Satya Sai, R. Codina and M. Vasquez, A general algorithm for compressible and incompressible flow – Part II, test on explicit form, *Int. J. Numer. Fluids*. 20 (1995) 887-913.

[2.25] P. Hansbo, Lagrangian incompressible flow computations in three dimensions by use of space-time finite elements, *Int. J. Numer. Fluids*. 20 (1995) 989-1001.

[2.26] S. Slijepčević, Computational modelling of non-Newtonian fluids based on the stabilised finite element method, *PhD thesis*. School of Engineering, University of Wales Swansea. (2003).

[2.27] T.C. Papanastasiou, Flow of materials with yield, *J. Rheol.* 31 (1987) 385-404.

[2.28] R.P. Chhabra, Bubbles, Drops and Particles in Non-Newtonian Fluids. (CRC Press, Florida, 1993)

[2.29] P.A.B. de Sampaio, A Petrov-Galerkin/Modified Operator Formulation for Convection-Diffusion Problems, *Int. J. Num. Meth. Engrg.* 30 (1990) 331-347.

[2.30] R.D. Cook, D.S. Malkus and M.E. Plesha, Concepts and applications of finite element analysis. 3rd Edition (Wiley, Toronto, 1989).

Chapter 3

Contact Modelling and Adaptive Remeshing

3.1 Introduction

In transit fluid-structure interaction problems, a Lagrangian mesh for the structure deforms with the structure and maintains a sharp definition of the moving boundary. The appropriate interface condition for the Lagrangian fluid mesh may be imposed by

$$\left(\boldsymbol{u}^s - \boldsymbol{u}^f\right)\cdot \boldsymbol{n} = 0 \qquad \text{on } \Gamma_c \tag{3.1}$$

where $\boldsymbol{n}$ is the unit outward normal to the interface (from the structure into the fluid), Γ_c denotes the contact interface, $\boldsymbol{u}^s$ and $\boldsymbol{u}^f$ represent the displacement field of the solid and fluid mesh at Γ_c, respectively. With constraint (3.1), the fluid nodes remain on the moving interface while permitting slip between the solid and the fluid meshes in the tangential direction. This condition is particularly useful in the analysis of free surface waves breaking against a solid wall or the transient response of water tank sloshing during transportation, where we need to let the fluid mesh slide along the structure. In liquid filling problems, fluid particles may contact each other. Newtonian fluids add the restriction that the fluid particles adhere, without slipping to the interface boundary, in such a situation we need to apply

$$\left(\boldsymbol{u}_1^f - \boldsymbol{u}_2^f\right)\cdot \boldsymbol{t} = 0 \qquad \text{on } \Gamma_c \tag{3.2}$$

where $\boldsymbol{t}$ is the unit tangent vector to the interface boundary Γ_c and $\boldsymbol{u}_1^f$ and $\boldsymbol{u}_2^f$ denote the displacement field of interacting fluid particles. Both constraints (3.1) and (3.2) represent the kinetic and kinematics condition on the contact interface and require a subsequent treatment of contact. In this chapter the finite element procedure of contact modelling for fluids on Lagrangian meshes is presented. The key aspect in computational contact mechanics is to apply the condition of impenetrability defined in equation (3.1). Many approaches have been proposed for imposing the constraints at the contact interface. There are three major methods, which are primarily considered and reviewed here.

- The penalty method.
- The Lagrange multiplier method
- The augmented Lagrangian method

The penalty method was developed in the early 1980's. Numerous research papers can be found in the literature. Hallquist *et al* [3.1][3.2] developed 2-D and 3-D slide-line contact algorithms, which were based on the penalty method. The closed form of the consistent linearization for the contact stiffness matrix of a 2-D deformed contact surface was derived by Wriggers *et al* [3.3] in 1985. Three dimensional frictional contact with a deformed body against a rigid surface was proposed by Peric and Owen [3.4]. In the implementation of the penalty method, a finite value of the penalty parameter is chosen to impose the contact constraint. The value of the penalty parameter has a significant effect on the solution accuracy and stability. If the penalty parameter is too small, the condition of impenetrability could be violated, on the other hand, a large penalty coefficient may deteriorate the condition number of the global stiffness matrix resulting in convergence problems in the implicit formulation and stability problem in the explicit analysis.

In contrast to the penalty method the Lagrange multipler approach ensures exact satisfaction of the required constraints, typical applications of the Lagrange multipler method can be found in reference [3.5]. However, in practice there are a number of disadvantages. Firstly, the number of unknown variables increases via the Lagrangian multiplers. Secondly, special care must be taken with the ordering of equations, since

the number of extra equations is often continually changing and extra equations associated with the Lagrangian multipliers have zero diagonals on the stiffness matrix, as in the standard mixed $\boldsymbol{u}$ - $\boldsymbol{p}$ Lagrangian flow formulation.

The objective of the augmented Lagrangian method, which was proposed by Simo and Laursen [3.6], is to minimize the disadvantage of the penalty method and Lagrangian multiplier method. It augments the Lagrangian with the penalty in the total potential energy of the system. By doing so, the difficulties associated with the 'solution ordering' in the Lagrangian multiplier approach are effectively removed since the contact tangential stiffness matrix is only related to the displacement field and is non-singular. Also the adopted penalty parameters need not be very large because the contact constraints are effectively satisfied via the Lagrangian multipliers. Nevertheless, we still have the disadvantage of the extra Lagrangian multiplier variables in the solution of the equations and two levels of iterative solution is required to obtain both displacement field and the Lagrangian multipliers. In the context of the thesis, the contact modelling is limited to the explicit analysis of the transient Stokes flow problem. The penalty method based discrete element contact algorithm is adopted to simulate fluid-structure or fluid-fluid particle contact. Since the time steps are small in the explicit time integration, the penalty method is well suited to enforcing inequality constraints for any class of problem, particularly for problems that involve large dynamic motion.

Transient Stokes flow problems involve a continually changing mesh configuration throughout the deformation process. The space-time finite element formulation introduced in the previous chapter has a built-in mechanism that can naturally accommodate an adaptive spatial mesh into the scheme, that consequently improves the accuracy of the finite element solution and enables it to carry on the simulation by overcoming excessive element distortions.

Over the years much progress has been achieved in the field of the adaptive remeshing. The rapid development is a consequence of the numerous researches on both accuracy error estimation [3.13][3.14][3.15][3.16] and transfer operators for evolving meshes [3.17][3.18]. At present, the formal structure and theoretical

foundation of adaptive remeshing for both fluid and solid structure problems are well understood.

The error estimation based adaptive remeshing was firstly proposed by Babuŝka and Rheinboldt [3.13] and later was further contributed by Zienkiewicz and Zhu [3.15][3.16], Oden *et al* [3.19] and Bank and Weiser [3.14]. The *a posteriori* error estimator was introduced by Zienkiewicz and Zhu [3.15] in 1987 to estimate the error in the energy norm. It provides the information required to generate or refine a mesh, which keeps the error within prescribed bounds. Several other error estimate criteria were proposed later. An error estimate based on plastic dissipation and the rate of plastic work has been developed by Peric *et al* [3.20], that is the most suitable error indicator for elasto-plastic or elasto-viscoplastic problems. The error estimate based on velocity gradients [3.21][3.22] was introduced for Navier-Stokes incompressible flow problems. It is important to remark that a criterion based on error in the velocity gradients concerns only the spatial discretization. Clearly such a criterion is well suited for space-time elements introduced in Chapter 2.

As the mesh is adaptive, with respect to an appropriate error estimator, the solution procedure cannot be re-computed from the initial state, but has to be continued from the previously computed state. The transformation of the state variables between two successive meshes needs to be properly carried out. Several important aspects involving the field values mapping were discussed in reference [3.17]. The focus of this chapter is on the weighted least squares mapping method, which significantly improves the mapping quality, compared to the background element mapping scheme.

The outline of this chapter is as follows: In section 3.2 the finite element procedure of contact modelling is presented, the variational principle of the elastic contact problem is introduced in section 3.2.1. In section 3.2.2 and 3.2.3 the contact force algorithms for node-to-facets of 2-D and 3-D contact elements are derived. The implementation of 2-D and 3-D contact elements with Coulomb friction is presented in section 3.2.4 and 3.2.5. In section 3.2.6 special contact cases; frictionless contact is introduced. The global and local contact search algorithm will be discussed in section 3.2.7. In section 3.3 the continuum adaptive remeshing procedure is introduced, in which the geometry entity related models definition is briefly presented in section 3.3.1. The error

estimation and mesh density prediction are described in section 3.3.2, with details of the implementation procedure. Section 3.3.3 reviews the mesh generator for unstructured meshes. Field values mapping operators, specially, the weighted least squares mapping, are discussed in section 3.3.4.

3.2 Contact modelling

The key issues related with enforcement of contact constraint in the finite element explicit formulation are listed and discussed in detail in the section.

- Description of contact phenomenon and the variational principle of the elastic contact problem.
- Derivation of contact force algorithm for 2-D and 3-D contact object with corresponding constitutive laws and in particular the Coulomb friction model.
- Selection of an effective, faster global and local contact search algorithm.

3.2.1 Computational contact mechanics

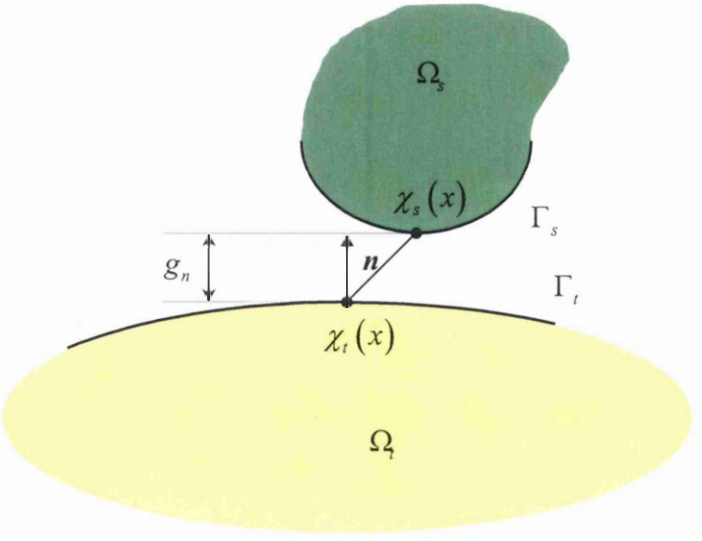


Figure 3.1 Contact between two bodies

Let us consider two bodies, one named the target body Ω_t and the other the impact body Ω_s, which may contact with each other during their deformation and movement. The boundaries of the target and the impact body will be denoted Γ_t and Γ_s respectively. The boundary of the target body Γ_t is characterized by an outward unit normal $\boldsymbol{n}$. At any stage of the deformation process corresponding to the configuration mapping of χ_t and χ_s of the target and impact bodies, the gap g_n is defined to separate the two bodies

$$g_n = \left(\chi_s(\boldsymbol{x}) - \chi_t(\boldsymbol{x})\right)\cdot \boldsymbol{n} \tag{3.3}$$

The kinematical constraint of the impenetrability between the two bodies can be written in the standard Kuhn-Tucker form as

$$g_n \geq 0 \qquad f_n \leq 0 \qquad f_n g_n = 0 \tag{3.4}$$

where f_n is the contact normal force acting on the impact body. By choosing the penalty method, the constraint condition (1) in equation (3.4) is relaxed, i.e. the penetration between the two bodies is assumed to be admissible. A linear relationship between the normal contact force and the normal gap is postulated as

$$\begin{aligned} f_n &= \varepsilon_n g_n \qquad if \;\; g_n < 0 \\ f_n &= 0 \qquad\qquad g_n \geq 0 \end{aligned} \tag{3.5}$$

where ε_n is called the normal stiffness or penalty value. In the context of the principle of virtual work, the virtual work of the contact forces imposed on the set of kinematically admissible gap rate can be added to the variational formulation (2.60) as

$$\bar{\varphi}(\boldsymbol{w},q) = \varphi(\boldsymbol{w},q) + \dot{\boldsymbol{g}}(\boldsymbol{w})\varepsilon \boldsymbol{g} \tag{3.6}$$

The variational function $\varphi(\boldsymbol{w},q)$ is minimized with respect to the virtual velocity $\boldsymbol{w}$, which leads to the expression

$$\begin{aligned} \frac{\partial \bar{\varphi}(\boldsymbol{w},q)}{\partial \boldsymbol{w}} &= \frac{\partial \varphi(\boldsymbol{w},q)}{\partial \boldsymbol{w}} + \varepsilon \boldsymbol{g} \frac{\partial \dot{g}}{\partial \boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{w}} \\ &= \frac{\partial \varphi(\boldsymbol{w},q)}{\partial \boldsymbol{w}} + \varepsilon \boldsymbol{g} \frac{\partial g}{\partial \boldsymbol{u}} = 0 \end{aligned} \tag{3.7}$$

The second term in the right side of equation (3.7) denotes the internal contact forces and the internal contact force vector for a single contact element can be defined as

$$\boldsymbol{f}_c^{\text{int}} = \boldsymbol{f}_{n,c}^{\text{int}} + \boldsymbol{f}_{t,c}^{\text{int}} = \varepsilon_n g_n \frac{\partial g_n}{\partial \boldsymbol{u}} + \varepsilon_t g_t \frac{\partial g_t}{\partial \boldsymbol{u}} \tag{3.8}$$

where ε_n, ε_t are the normal and tangential penalty values, and subscripts n and t are referred to the normal and tangential directions. The calculation of the normal and tangential force vector for 2-D and 3-D problems will be introduced in sections 3.2.2 and 3.2.3.

3.2.2 Contact forces for 2-D node-to-facet

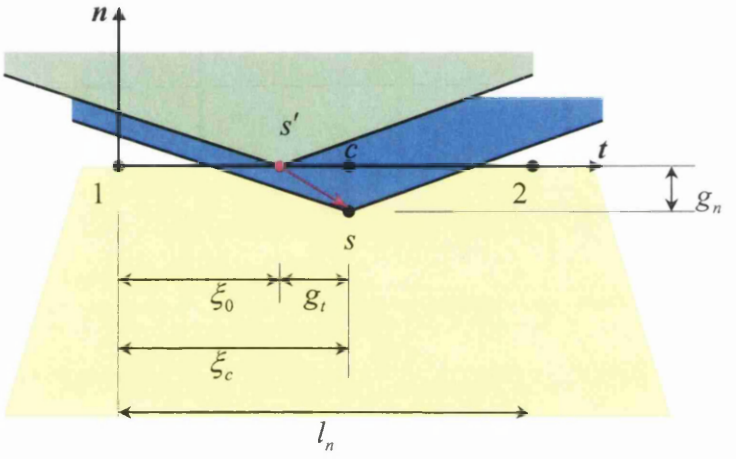


Figure 3.2 Node to facet contact

In Figure 3.2 it is assumed that an impact node from previous position s' moved to s at the current time step, where $\boldsymbol{n}$ and $\boldsymbol{t}$ denote the normal and tangential vectors of the target facet 1-2. l_0 and l_n are the previous and current lengths of the target facet. The normal gap g_n and tangential gap g_t are given by

$$g_n = (\boldsymbol{x}_s - \boldsymbol{x}_1) \cdot \boldsymbol{n} = (\boldsymbol{x}_s - \boldsymbol{x}_c) \cdot \boldsymbol{n} \tag{3.9}$$

$$g_t = \xi_c l_n - \xi_0 l_0 \tag{3.10}$$

where $\boldsymbol{x}_s$, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are the spatial position of the impact node and the target nodes at the current time step, the three nodes form a contact element. $\boldsymbol{x}_c$ denotes the position of node s projected onto the target facet 1-2. The local coordinates of the contact location is defined as

$$\xi_c = \frac{1}{l_n}(\boldsymbol{x}_s - \boldsymbol{x}_1)\cdot \boldsymbol{t} \tag{3.11}$$

and

$$\boldsymbol{t} = (\boldsymbol{x}_2 - \boldsymbol{x}_1)\frac{1}{l_n} \qquad \boldsymbol{n} = \boldsymbol{e}_3 \times \boldsymbol{t} \tag{3.12}$$

The directional variation of $\boldsymbol{t}$ and $\boldsymbol{n}$ are defined as

$$\delta \boldsymbol{t} = \frac{1}{l_n}(\boldsymbol{I} - \boldsymbol{t}\otimes\boldsymbol{t})(\delta \boldsymbol{u}_2 - \delta \boldsymbol{u}_1) \tag{3.13}$$

$$\delta \boldsymbol{n} = -\frac{1}{l_n}(\boldsymbol{t}\otimes\boldsymbol{n})(\delta \boldsymbol{u}_2 - \delta \boldsymbol{u}_1) \tag{3.14}$$

The variation of the normal gap g_n in equation (3.9) can be derived with the aid of equations (3.11)-(3.14)

$$\begin{aligned}\delta g_n &= \boldsymbol{n}\cdot(\delta \boldsymbol{u}_s - \delta \boldsymbol{u}_1) + \delta \boldsymbol{n}\cdot(\boldsymbol{x}_s - \boldsymbol{x}_1)\\ &= \boldsymbol{n}\cdot(\delta \boldsymbol{u}_s - \delta \boldsymbol{u}_1) - \frac{1}{l_n}(\boldsymbol{x}_s - \boldsymbol{x}_1)\cdot(\boldsymbol{t}\otimes\boldsymbol{n})(\delta \boldsymbol{u}_2 - \delta \boldsymbol{u}_1)\\ &= \boldsymbol{n}\cdot\left[\delta \boldsymbol{u}_s - (1-\xi_c)\delta \boldsymbol{u}_1 - \xi_c \delta \boldsymbol{u}_2\right]\end{aligned} \tag{3.15}$$

Now the element internal normal contact forces $\boldsymbol{f}_{n,c}^{\text{int}}$ can be written as

$$\boldsymbol{f}_{n,c}^{\text{int}} = \varepsilon_n g_n \begin{bmatrix} \boldsymbol{n} \\ -(1-\xi_c)\boldsymbol{n} \\ -\xi_c \boldsymbol{n} \end{bmatrix} = \varepsilon_n g_n \boldsymbol{N}_s \tag{3.16}$$

and

$$\boldsymbol{N}_s = \begin{bmatrix} \boldsymbol{n} \\ -N_1(\xi_c)\boldsymbol{n} \\ -N_2(\xi_c)\boldsymbol{n} \end{bmatrix} = \begin{bmatrix} \boldsymbol{n} \\ -(1-\xi_c)\boldsymbol{n} \\ -\xi_c \boldsymbol{n} \end{bmatrix} \tag{3.17}$$

The variation of the tangential gap g_t is derived using equation (3.10), where ξ_0 is the local coordinates at the previous time step. One might think of defining g_t differently with current length l_n in (3.10). However it turns out that the resulting tangent stiffness matrix is then non-symmetric. With the aid of equation (3.13) and (3.14)

$$\begin{aligned}\delta g_t &= l_0 \delta \xi_c \\ &= \frac{l_0}{l_n}\left[(\delta \boldsymbol{u}_s - \delta \boldsymbol{u}_1)\cdot \boldsymbol{t} + (\boldsymbol{x}_s - \boldsymbol{x}_1)\cdot \delta \boldsymbol{t} - \frac{1}{l_n}\delta l_n (\boldsymbol{x}_s - \boldsymbol{x}_1)\cdot \boldsymbol{t}\right]\end{aligned} \tag{3.18}$$

and

$$\delta l_n = \frac{1}{l_n}(\boldsymbol{x}_2 - \boldsymbol{x}_1)\cdot(\delta \boldsymbol{u}_2 - \delta \boldsymbol{u}_1) \tag{3.19}$$

Substitute δl_n and $\delta \boldsymbol{t}$ into equation (3.18) then the equation (3.18) can be re-arranged as

$$\begin{aligned}\delta g_t = \frac{l_0}{l_n}\Bigg[&(\delta \boldsymbol{u}_s - \delta \boldsymbol{u}_1)\cdot \boldsymbol{t} - \frac{1}{l_n}(\boldsymbol{x}_s - \boldsymbol{x}_1)\cdot(\boldsymbol{t}\otimes \boldsymbol{t})(\delta \boldsymbol{u}_2 - \delta \boldsymbol{u}_1) \\ &+ \frac{1}{l_n}(\boldsymbol{x}_s - \boldsymbol{x}_1)\cdot(\delta \boldsymbol{u}_2 - \delta \boldsymbol{u}_1) - \frac{\xi_c}{l_n}(\boldsymbol{x}_2 - \boldsymbol{x}_1)\cdot(\delta \boldsymbol{u}_2 - \delta \boldsymbol{u}_1)\Bigg] \\ = \frac{l_0}{l_n}\Big\{&\left[\delta \boldsymbol{u}_s - (1-\xi_c)\delta \boldsymbol{u}_1 - \xi_c \delta \boldsymbol{u}_2\right]\cdot \boldsymbol{t} + \frac{g_n}{l_n}(\delta \boldsymbol{u}_2 - \delta \boldsymbol{u}_1)\cdot \boldsymbol{n}\Big\}\end{aligned} \tag{3.20}$$

The element internal tangential contact forces $\boldsymbol{f}_{t,c}^{\text{int}}$ can be finally defined as

$$\begin{aligned}\boldsymbol{f}_{t,c}^{\text{int}} &= \varepsilon_t g_t \frac{l_0}{l_n}\left\{\begin{bmatrix}\boldsymbol{t}\\ -(1-\xi_c)\boldsymbol{t}\\ -\xi_c \boldsymbol{t}\end{bmatrix} + \frac{g_n}{l_n}\begin{bmatrix}0\\ -\boldsymbol{n}\\ \boldsymbol{n}\end{bmatrix}\right\} \\ &= \varepsilon_t g_t \frac{l_0}{l_n}\left\{\boldsymbol{T}_s + \frac{g_n}{l_n}\boldsymbol{N}_m\right\} = f_t \frac{l_0}{l_n}\left\{\boldsymbol{T}_s + \frac{g_n}{l_n}\boldsymbol{N}_m\right\}\end{aligned} \tag{3.21}$$

where

$$\boldsymbol{T}_s = \begin{bmatrix}\boldsymbol{t}\\ -N_1(\xi_c)\boldsymbol{t}\\ -N_2(\xi_c)\boldsymbol{t}\end{bmatrix} = \begin{bmatrix}\boldsymbol{t}\\ -(1-\xi_c)\boldsymbol{t}\\ -\xi_c \boldsymbol{t}\end{bmatrix} \qquad \boldsymbol{N}_m = \begin{bmatrix}0\\ -\boldsymbol{n}\\ \boldsymbol{n}\end{bmatrix} \tag{3.22}$$

The second term on the right side of equation (3.21) is caused by change of the length of the target facet. If the target facet is a rigid one, then $l_n = l_0$ and equation (3.21) can be simply replaced by

$$\boldsymbol{f}_{t,c}^{\text{int}} = \varepsilon_t g_t \boldsymbol{T}_s \qquad 3.23$$

Remark:

(1) Equation (3.16) illustrates that the contact normal forces $\varepsilon_n g_n$ on nodes s along the local normal direction is simply balanced by the reaction forces $-N_1(\xi_c)\varepsilon_n g_n$ and $-N_2(\xi_c)\varepsilon_n g_n$ on nodes 1, 2 of the target facet using the corresponding nodal shape functions. Then those local nodal forces are projected onto the global system using a transformation vector $\boldsymbol{n}$. The same rule is applied to the definition of the contact tangential nodal forces, if a rigid target segment is assumed, shown in equation (3.23). The techniques of computing contact forces can be extended to any shape of the target facet of 2d and 3d cases, once the local coordinates of the contact point on the target facet are defined, the global nodal contact forces can be easily defined.

(2) The tangential contact force $\boldsymbol{f}_{t,c}^{\text{int}}$ is called the 'sticking friction' force in some references [3.7], which is proportional to the tangential gap. If $\boldsymbol{f}_{t,c}^{\text{int}}$ is set to zero, it is a frictionless contact case. From frictionless to complete 'sticking friction', there is a type of sliding friction, which is usually defined by the Coulomb friction law. Numerical models of linear and non-linear friction will be discussed in detail in sections 3.2.4 and 3.2.5.

3.2.3 Contact forces for 3-D node-to-facet

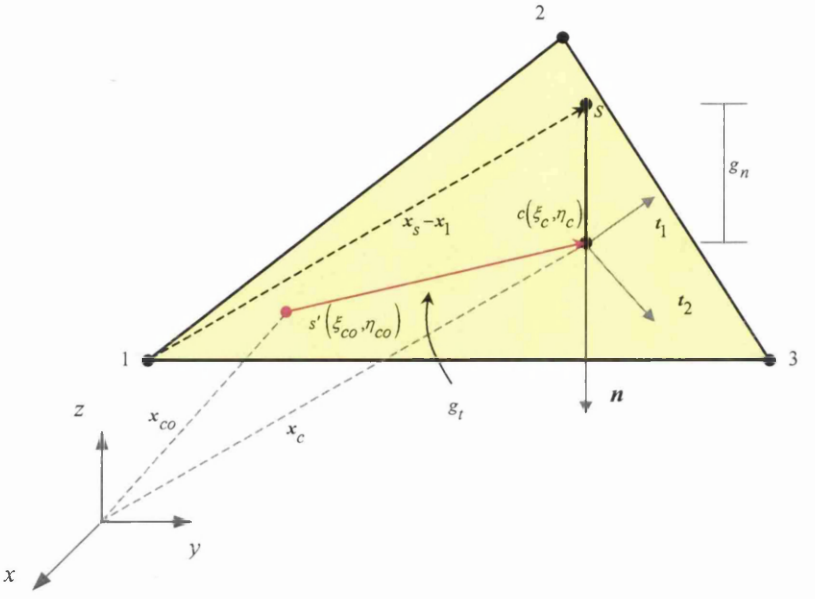


Figure 3.3 Node-to-facet contact with three noded-facet

For a general three-dimensional analysis, the contact surface is two-dimensional. In Figure 3.3 this surface is assumed to be a three noded surface, which may be associated with an underlying tetrahedral element or 3-noded rigid facet. The geometry of the target facet can be defined by a set of unit vector base $\{\boldsymbol{n},\ \boldsymbol{t}_1,\ \boldsymbol{t}_2\}$ as

$$\boldsymbol{t}_1 = \frac{(\boldsymbol{x}_2 - \boldsymbol{x}_1)}{\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|} \qquad \boldsymbol{e}_2 = \frac{(\boldsymbol{x}_3 - \boldsymbol{x}_1)}{\|\boldsymbol{x}_3 - \boldsymbol{x}_1\|}$$

$$\boldsymbol{n} = \boldsymbol{t}_1 \times \boldsymbol{e}_2 \qquad \boldsymbol{t}_2 = \boldsymbol{n} \times \boldsymbol{t}_1 \tag{3.24}$$

where $\boldsymbol{n}$ is a normal vector to the target facet at contact point $\boldsymbol{x}_c$. Assuming the impact node moved from the previous contact position s' to s at the current time step. The normal penetration gap g_n and the tangential gap vector $\boldsymbol{g}_t$ are given by

$$g_n = (\boldsymbol{x}_s - \boldsymbol{x}_1) \cdot \boldsymbol{n} = (\boldsymbol{x}_s - \boldsymbol{x}_c) \cdot \boldsymbol{n} \tag{3.25}$$

$$\boldsymbol{g}_t = \begin{Bmatrix} g_{t1} \\ g_{t2} \end{Bmatrix} = \begin{Bmatrix} (\boldsymbol{x}_s - \boldsymbol{x}_1) \cdot \boldsymbol{t}_1 \\ (\boldsymbol{x}_s - \boldsymbol{x}_1) \cdot \boldsymbol{t}_2 \end{Bmatrix} - \boldsymbol{g}_{to} \tag{3.26}$$

where $\boldsymbol{g}_{to}$ is the previous tangential contact gap vector in the target facet. According to the remark made in 3.2.2, the contact normal and tangential forces for a 3-noded facet can be simply defined as

$$\boldsymbol{f}_{n,c}^{\text{int}} = \varepsilon_n g_n \boldsymbol{n} \boldsymbol{N}_{3c} \qquad \boldsymbol{N}_{3c} = \begin{bmatrix} 1 \\ -N_1(\xi_c, \eta_c) \\ -N_2(\xi_c, \eta_c) \\ -N_3(\xi_c, \eta_c) \end{bmatrix} \tag{3.27}$$

$$\boldsymbol{f}_{t,c}^{\text{int}} = \left(\varepsilon_t g_{t_1} \boldsymbol{t}_1 + \varepsilon_t g_{t_2} \boldsymbol{t}_2\right) \boldsymbol{N}_{3c} \tag{3.28}$$

where the local coordinates related to the nodal shape functions in equation (3.27) and (3.28) are defined using area coordinates

$$\begin{aligned} N_1(\xi_c, \eta_c) &= \xi_c = \frac{A_{c23}}{A} \\ N_2(\xi_c, \eta_c) &= \eta_c = \frac{A_{c31}}{A} \\ N_3(\xi_c, \eta_c) &= 1 - \xi_c - \eta_c \end{aligned} \tag{3.29}$$

where A is the target facet area, A_{c23} is the area defined by contact point $\boldsymbol{x}_c$ and nodal position $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$. A_{c31} is the area defined by contact point $\boldsymbol{x}_c$ and nodal position $\boldsymbol{x}_3$ and $\boldsymbol{x}_1$. The position of contact point $\boldsymbol{x}_c$ is set by equation (3.25) as

$$\boldsymbol{x}_c = \boldsymbol{x}_s + g_n \boldsymbol{n} \tag{3.30}$$

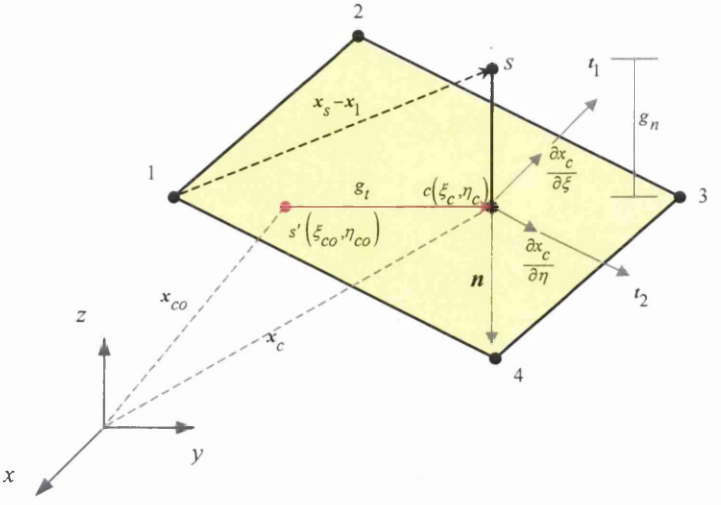


Figure 3.4 Node-to-facet contact with four noded-facet

If the target facet is a 4-noded facet, which may be associated with a hexahedral solid element or 4-noded rigid facet, the definition of local coordinate (ξ_c, η_c) is not a trivial matter. Assuming a contact point c is located on the target facet, which is closest to node s, as shown in Figure 3.4, and defined by a bilinear interpolation function as

$$\boldsymbol{x}_c = \sum N_i(\xi_c, \eta_c)\boldsymbol{x}_i \tag{3.31}$$

$$N_i(\xi_c, \eta_c) = \frac{1}{4}(1+\xi_c\xi_i)(1+\eta_c\eta_i) \tag{3.32}$$

where ξ_i and η_i take on their nodal values at $(\pm 1, \pm 1)$ and $\boldsymbol{x}_i$ is the nodal coordinate of the i^{th} node on the target segment. The terms (ξ_c, η_c) are assumed to be the contact point local coordinates which need to be defined. The tangential direction at position $\boldsymbol{x}_c$ must be orthogonal to the normal vector $(\boldsymbol{x}_s - \boldsymbol{x}_c)$, i.e. the local coordinates (ξ_c, η_c) must satisfy

$$\begin{aligned} \frac{\partial \boldsymbol{x}_c(\xi_c, \eta_c)}{\partial \xi} \cdot \left(\boldsymbol{x}_s - \boldsymbol{x}_c(\xi_c, \eta_c)\right) = 0 \\ \frac{\partial \boldsymbol{x}_c(\xi_c, \eta_c)}{\partial \eta} \cdot \left(\boldsymbol{x}_s - \boldsymbol{x}_c(\xi_c, \eta_c)\right) = 0 \end{aligned} \tag{3.33}$$

Equation (3.33) was given by Hallquist et al [3.2] and can be readily solved using the Newton-Raphson iteration method, with initial estimates for (ξ_c, η_c). Once (ξ_c, η_c) are defined by equation (3.33), the tangential and normal directions at contact point $\boldsymbol{x}_c$ can be set as

$$\boldsymbol{t}_1 = \frac{\partial \boldsymbol{x}_c}{\partial \xi} \Big/ \left\| \frac{\partial \boldsymbol{x}_c}{\partial \xi} \right\| \qquad \boldsymbol{t}_2 = \frac{\partial \boldsymbol{x}_c}{\partial \eta} \Big/ \left\| \frac{\partial \boldsymbol{x}_c}{\partial \eta} \right\|$$

$$\boldsymbol{n} = \boldsymbol{t}_1 \times \boldsymbol{t}_2 \tag{3.34}$$

The contact normal and tangential forces for the 4-noded facet can be defined as

$$\boldsymbol{f}_{n,c}^{\text{int}} = \varepsilon_n g_n \boldsymbol{n} \boldsymbol{N}_{4c} \qquad \boldsymbol{N}_{4c} = \begin{bmatrix} 1 \\ -N_1(\xi_c, \eta_c) \\ -N_2(\xi_c, \eta_c) \\ -N_3(\xi_c, \eta_c) \\ -N_4(\xi_c, \eta_c) \end{bmatrix} \tag{3.35}$$

$$\boldsymbol{f}_{t,c}^{\text{int}} = \left(\varepsilon_t g_{t_1} \boldsymbol{t}_1 + \varepsilon_t g_{t_2} \boldsymbol{t}_2\right) \boldsymbol{N}_{4c} \tag{3.36}$$

3.2.4 Coulomb friction forces for 2-D contact element

The tangential forces defined in equation (3.21) or (3.23) are the linear elastic type forces, or adherence forces. When sliding friction is considered, the Coulomb friction law is introduced by several authors [3.8][3.9]. A typical Amontons and Coulomb law of friction is summarized by K. Hashimoto *et al* [3.10] as follows:

(1) The friction is independent of the apparent area of the two contact bodies.

(2) The friction force is proportional to the normal contact force between them.

(3) The kinetic friction is almost independent of the speed of sliding.

In the Coulomb friction model, the tangential friction force is governed by the Coulomb friction law, or called the yielding function as

$$\phi(f_t) = |f_t| - c|f_n| = 0 \tag{3.37}$$

where c is the friction coefficient, which is dependent on the material medium and roughness of the contact surface. Here a framework for the plasticity theory of friction is reviewed for the 2-D case. The incremental tangential gap Δg_t defined by equation (3.10) may be decomposed into an elastic component and plastic or sliding component as

$$\Delta g_t = \Delta g_t^e + \Delta g_t^p \tag{3.38}$$

Following the elastic constitutive law the total tangential force is defined by

$$f_t^{n+1} = f_t^n + \varepsilon_t \Delta g_t^e = f_t^n + \varepsilon_t \left(\Delta g_t - \Delta g_t^p \right) = f_{t,trial}^{n+1} - \varepsilon_t \Delta g_t^p \qquad 3.39$$

where f_t^n is the previous tangential force, $f_{t,trial}^{n+1}$ is the trial tangential force in the current step. The evolution law for the sliding component is set by a non-associated flow rule as

$$\Delta g_t^p = \dot{\lambda} \frac{\partial \Psi}{\partial f_t} \qquad 3.40$$

where Ψ is the sliding potential and its derivative $\frac{\partial \Psi}{\partial f_t}$ defines the direction of sliding, they are set by

$$\Psi(f_t) = |f_t| \qquad 3.41$$

and

$$\frac{\partial \Psi}{\partial f_t} = \frac{f_t}{|f_t|} = sign(f_t) \qquad 3.42$$

The direction of friction sliding is the same as the direction of the tangential force. If an associated flow rule was adopted, the plastic sliding would occur in the normal direction of the target facet, as shown in Figure 3.5, this would be physically unrealistic [3.7]

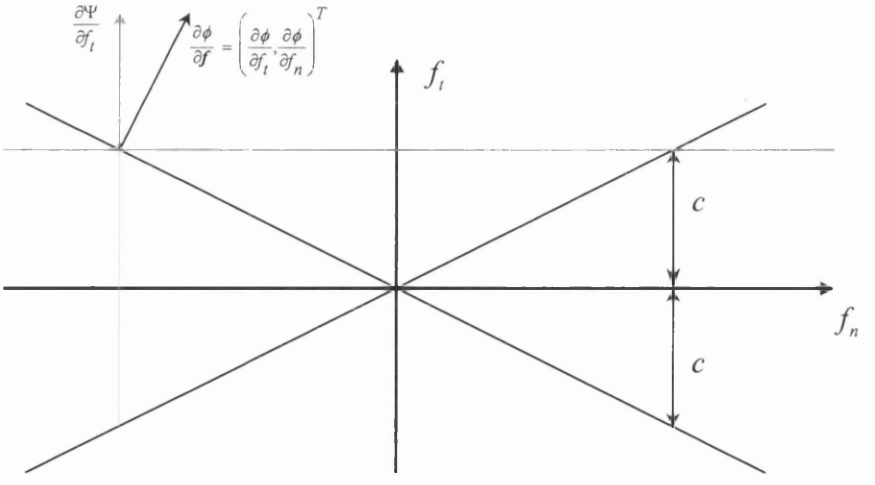


Figure 3.5 Coulomb friction model

The plastic multiplier $\dot{\lambda}$ satisfies the complementary conditions

$$\phi(f_t) \le 0 \qquad \dot{\lambda} \ge 0 \qquad \dot{\lambda}\phi(f_t) = 0 \tag{3.43}$$

Substituting equation (3.39) and (3.40) into the yielding function (3.37) the plastic multiplier $\dot{\lambda}$ is given by

$$\dot{\lambda} = \frac{\left|f_{t,trial}^{n+1}\right| - c\left|f_n\right|}{\varepsilon_t} \tag{3.44}$$

The tangential friction force f_t^{n+1} and incremental sliding gap Δg_t^p along the tangential direction $\boldsymbol{t}$ can be written as

$$f_t^{n+1} = c\left|f_n\right| sign\left(f_{t,trial}^{n+1}\right) \tag{3.45}$$

$$\Delta g_t^p = \dot{\lambda} sign\left(f_{t,trial}^{n+1}\right) \tag{3.46}$$

Substituting equation (3.45) into (3.21) the element internal tangential contact forces $f_{t,c}^{\text{int}}$ for the sliding friction are determined. The implementation of the tangential friction force algorithm for the 2-D contact element is summarised in Figure 3.6.

---

(1) Update configuration

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \dot{\boldsymbol{u}} \cdot \Delta t$$

(2) Evaluate trial tangential forces

$$f_{t,trial}^{n+1} = f_t^n + \varepsilon_t \Delta g_t$$

(3) Check plastic sliding condition

if $\phi\left(f_{t,trial}^{n+1}\right) = \left|f_{t,trial}^{n+1}\right| - c\left|f_n\right| \le 0$ then

$$f_t^{n+1} = f_{t,trial}^{n+1}$$

Else

$$\dot{\lambda} = \left(\left|f_{t,trial}^{n+1}\right| - c\left|f_n\right|\right) / \varepsilon_t$$

(4) Update local friction forces

$$f_t = c\left|f_n\right| sign\left(f_{t,trial}^{n+1}\right)$$

$$g_t^{n+1} = g_t^n + \dot{\lambda} sign\left(f_{t,trial}^{n+1}\right)$$

Endif

(5) Update global friction forces

$$\boldsymbol{f}_{t,c}^{\text{int}} = f_t \frac{l_0}{l_n}\left(\boldsymbol{T}_s + \frac{g_n}{l_n}\boldsymbol{N}_m\right) \quad (3.21)$$

or $\boldsymbol{f}_{t,c}^{\text{int}} = f_t \boldsymbol{T}_s \quad (3.23)$

Figure 3.6 2-D Contact tangential force update procedure

3.2.5 Coulomb friction forces for 3-D contact element

In 3-D the incremental tangential gap vector $\Delta \boldsymbol{g}_t$ given by equation (3.26) is decomposed into elastic and plastic components as

$$\Delta \boldsymbol{g}_t = \Delta \boldsymbol{g}_t^e + \Delta \boldsymbol{g}_t^p \tag{3.47}$$

The tangential force vector $\boldsymbol{f}_t$ is now given by

$$\boldsymbol{f}_t = \begin{Bmatrix} f_{t1} \\ f_{t2} \end{Bmatrix} = \boldsymbol{f}_t^n + \varepsilon_t \left(\Delta \boldsymbol{g}_t - \Delta \boldsymbol{g}_t^p \right) = \boldsymbol{f}_{t,trial}^{n+1} - \varepsilon_t \Delta \boldsymbol{g}_t^p \tag{3.48}$$

The yielding function defined in equation (3.37) is altered to give

$$\phi(\boldsymbol{f}_t) = \|\boldsymbol{f}_t\| - c|f_n| = 0 \tag{3.49}$$

The incremental plastic sliding $\Delta \boldsymbol{g}_t^p$ is defined normal to the cylinder $\|\boldsymbol{f}_t\| = \text{constant}$ by a non-associative flow rule as

$$\Delta \boldsymbol{g}_t^p = \dot{\lambda} \frac{\partial \Psi}{\partial \boldsymbol{f}_t} = \dot{\lambda} \frac{\boldsymbol{f}_t}{\|\boldsymbol{f}_t\|} \tag{3.50}$$

where the sliding potential $\Psi = \|\boldsymbol{f}_t\|$. Substituting equation (3.50) into equation (3.48) gives

$$\begin{aligned} \boldsymbol{f}_t &= \boldsymbol{f}_{t,trial}^{n+1} - \varepsilon_t \dot{\lambda} \frac{\boldsymbol{f}_t}{\|\boldsymbol{f}_t\|} \\ &= \left(1 - \varepsilon_t \dot{\lambda} / \|\boldsymbol{f}_t\|\right) \boldsymbol{f}_{t,trial}^{n+1} = \alpha \boldsymbol{f}_{t,trial}^{n+1} \end{aligned} \tag{3.51}$$

Equation (3.51) is based on the plastic radial return assumption [3.4] i.e. plastic flow directions at a trial stress point $\boldsymbol{f}_{t,trial}^{n+1}$ and at yielding surface $\boldsymbol{f}_t$ are the same. The plastic multiplier $\dot{\lambda}$ can be obtained via the satisfaction of the complementary condition (3.43) and the yielding criteria $\phi(\boldsymbol{f}_t) = 0$, from which

$$\dot{\lambda} = \left(\|\boldsymbol{f}_{t,trial}^{n+1}\| - c|f_n| \right) / \varepsilon_t \tag{3.52}$$

The implementation of the tangential friction force algorithm for the 3-D contact element is presented in Figure 3.7.

(1) Update configuration

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \dot{\boldsymbol{u}} \cdot \Delta t$$

(2) Evaluate trial tangential forces

$$\boldsymbol{f}_{t,trial}^{n+1} = \boldsymbol{f}_t^n + \varepsilon_t \Delta \boldsymbol{g}_t$$

(3) Check plastic sliding condition

if $\phi\left(\boldsymbol{f}_{t,trial}^{n+1}\right) = \left\|\boldsymbol{f}_{t,trial}^{n+1}\right\| - c\left|f_n\right| \le 0$ then

$$\boldsymbol{f}_t^{n+1} = \begin{Bmatrix} f_{t1} \\ f_{t2} \end{Bmatrix} = \boldsymbol{f}_{t,trial}^{n+1}$$

Else

$$\dot{\lambda} = \left(\left\|\boldsymbol{f}_{t,trial}^{n+1}\right\| - c\left|f_n\right|\right)/\varepsilon_t \qquad \boldsymbol{T} = \frac{\boldsymbol{f}_{t,trial}}{\left\|\boldsymbol{f}_{t,trial}\right\|}$$

(4) Update local friction forces

$$\boldsymbol{f}_t^{n+1} = \begin{Bmatrix} f_{t1} \\ f_{t2} \end{Bmatrix} = c\left|f_n\right|\boldsymbol{T}$$

$$\boldsymbol{g}_t^{p,n+1} = \boldsymbol{g}_t^{p,n} + \dot{\lambda}\boldsymbol{T} \qquad \text{Update sliding distance}$$

Endif

(6) Update global friction forces

$$\boldsymbol{f}_{t,c}^{\text{int}} = \left(f_{t1}\boldsymbol{t}_1 + f_{t2}\boldsymbol{t}_2\right)\boldsymbol{N}_{3c(or\,4c)} \quad (3.28) \text{ or } (3.36)$$

Figure 3.7 3-D Contact tangential force update procedure

It has to be mentioned that a friction hardening model was developed by E.A. de Souza Neto *et al* [3.9] in 1993, to simulate the frictional behaviour of coated steel, where the evolution of surface wear become particularly important in the definition of the frictional behaviour. The friction coefficient c is no longer a constant value, and is dependent on the density of frictional work expended on the contact surface considered. Analogous to the classical work hardening elasto-plasticity theory, the yielding function is defined as

$$\phi\left(\boldsymbol{f}_t, w\right) = \left\|\boldsymbol{f}_t\right\| - c\left(w\right)\left|f_n\right| = 0 \qquad 3.53$$

where w is the density of frictional work expended on the contact point considered, its evolution equation is given by

$$\Delta w = \boldsymbol{f}_t \Delta \boldsymbol{g}_t^p = \left\| \boldsymbol{f}_t \right\| \dot{\lambda} = c(w) |f_n| \dot{\lambda} \tag{3.54}$$

Instead of solving equations (3.44) or (3.52) to obtain the plastic multiplier $\dot{\lambda}$, we must solve the following coupled residual equations to obtain plastic multiplier $\dot{\lambda}$ and incremental frictional work density Δw as

$$\begin{aligned} r_1 &= \left\| \boldsymbol{f}_{t,trial}^{n+1} \right\| - \varepsilon_t \dot{\lambda} - c(w_{n+1}) |f_n| = 0 \\ r_2 &= w_{n+1} - w_n - c(w_{n+1}) |f_n| \dot{\lambda} = 0 \end{aligned} \tag{3.55}$$

The resulting system of non-linear algebraic equation (3.55) can be solved by the standard Newton-Raphson iteration method as

$$\begin{Bmatrix} \dot{\lambda} \\ \Delta w \end{Bmatrix}_{i+1} = \begin{Bmatrix} \dot{\lambda} \\ \Delta w \end{Bmatrix}_i - \begin{bmatrix} -\varepsilon_t & -\dfrac{\partial c}{\partial w} |f_n| \\ -c|f_n| & 1 - \dfrac{\partial c}{\partial w} |f_n| \dot{\lambda} \end{bmatrix}^{-1} \begin{Bmatrix} \delta r_1 \\ \delta r_2 \end{Bmatrix} \tag{3.56}$$

and

$$w_{n+1} = w_n + \Delta w_{i+1} \tag{3.57}$$

where the derivative of the friction coefficient with respective to the friction work is given by the frictional hardening curve $c = c(w)$. For electrogalvanised steel sheet (EG) it is defined by a polynomial function with w measured in *kN/cm*

$$\begin{aligned} c(w) = & -0.4096 \times 10^{-6} w^5 + 0.2890 \times 10^{-4} w^4 - 0.8212 \times 10^{-3} w^3 \\ & + 0.1035 \times 10^{-1} w^2 - 0.3148 \times 10^{-1} w + 0.1568 \end{aligned} \tag{3.58}$$

3.2.6 Frictionless contact

The frictionless contact is a special contact case. In the frictionless contact, the contact forces along the tangential sliding directions of a facet are kept to zero and impenetrability constraint is applied on the normal direction of the facet. On the numerical modelling of steep forced water waves against a rigid tank wall, a special contact treatment is set for the impact node. The global velocities on the impact and

contact nodes are projected onto a local coordinate system, which is defined by the facet geometry as:

$$\dot{\boldsymbol{u}}_s^l = \begin{Bmatrix} \dot{u}_{n,s} \\ \dot{u}_{t,s} \end{Bmatrix} = \begin{bmatrix} \boldsymbol{n}^T \\ \boldsymbol{t}^T \end{bmatrix} \dot{\boldsymbol{u}}_s \tag{3.59a}$$

$$\dot{\boldsymbol{u}}_c^l = \begin{Bmatrix} \dot{u}_{n,c} \\ \dot{u}_{t,c} \end{Bmatrix} = \begin{bmatrix} \boldsymbol{n}^T \\ \boldsymbol{t}^T \end{bmatrix} \dot{\boldsymbol{u}}_c \tag{3.59b}$$

and

$$\dot{u}_{n,s} = \dot{u}_{n,c} - g_n/\Delta t \quad if\ g_n < 0 \tag{3.60}$$

$$\boldsymbol{f}_{t,c}^{\text{int}} = 0 \tag{3.61}$$

where $\dot{\boldsymbol{u}}_s^l$ and $\dot{\boldsymbol{u}}_s$ are the local and global velocities on the impact node, $\dot{\boldsymbol{u}}_c^l$ and $\dot{\boldsymbol{u}}_c$ are the local and global velocities on the contact node of the facet, as seen in Figure 3.2. $\dot{u}_{n,s}$ and $\dot{u}_{n,c}$ are the normal velocity components, $\dot{u}_{n,s}$ is adjusted according to the impenetrability constraints in the normal direction of the facet. Once contact is established, the normal velocity $\dot{u}_{n,s}$ is maintained equal to the normal velocity of the contact node, $\dot{u}_{n,c}$. Obviously, the impact nodes applied with frictionless contact are only assigned to the deformed mesh domain.

3.2.7 Contact detection

One of the most important aspects of contact modelling is the development of an effective detection procedure for monitoring contact between large numbers of discrete objects. The alternating digital tree (ADT) algorithm [3.11] is a spatial global search algorithm based on space-cell subdivision and incorporating a tree data storage structure and possesses significant computational advantage. The algorithm was later further developed by Feng and Owen [3.12] as an augmented spatial digital tree (ASDT) for problems involving simple geometric rectangular objects, which is generally achieved by representing any arbitrarily shaped object with an axis aligned bounding box. The ASDT algorithm uses only the lower corner vertex to represent a rectangular object with the upper corner vertex serving as the augmented information. Consequently it gives a better-balanced tree and reduces the CPU time of contact

detection significantly. The contact detection procedure with ADT or ASDT algorithm can be divided into four stages.

1. *Body location mapping.*
2. *Space bisection.*
3. *Bounding box intersection.*
4. *Local contact resolution.*

The procedure of stages 1-3 comprises the global spatial search algorithm, in which all the objects are approximated by rectangles and each rectangle is reduced to a point in a higher (2n)-dimensional space. Based on this simplification a potential list of contact target facets for each contactor node is identified. In stage four, the local contact search algorithm is employed to identify the closest target facet to the contactor node under consideration.

3.2.7.1 Body location mapping

Each body, which can be a contactor node or a target facet, is circumscribed with an axis aligned bounding box, whose edges are parallel to the axis of the global coordinate system and is possibly further extended by a buffer zone, as shown in Figure 3.8

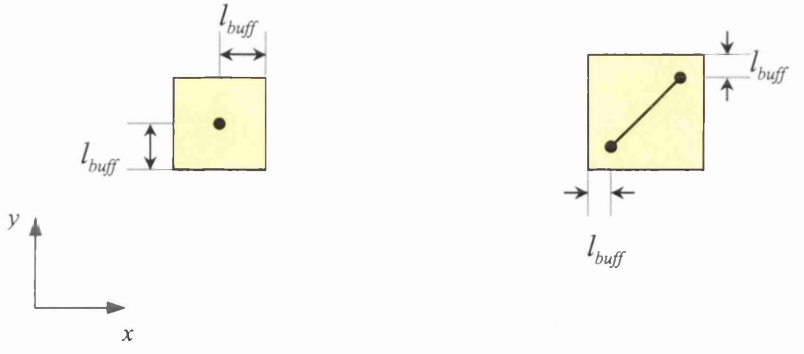


Figure 3.8 Bounding box definition for 2d, a contactor node and a target facet.

The body i, which denotes the contactor node or the target facet, is defined by two corner points $\boldsymbol{x}_{i,\min}$ and $\boldsymbol{x}_{i,\max}$ of its rectangular bounding box in an n-dimensional Euclidean space $\boldsymbol{R}^n$ and reduced as a single point in 2n-dimensional space $\boldsymbol{R}^{2n}$ via spatial mapping $L:\boldsymbol{R}^n \rightarrow \boldsymbol{R}^{2n}$. This point is termed the *representative point* of the rectangular bounding box and is defined by two characteristic corner points as

$$\boldsymbol{p}_i = \left[x^1_{i,\min},\ldots,x^n_{i,\min},x^1_{i,\max},\ldots,x^n_{i,\max}\right] \qquad 3.62$$

For example, a segment in $\boldsymbol{R}^1$ space is represented as a single point in $\boldsymbol{R}^2$ space as illustrated in Figure 3.9.

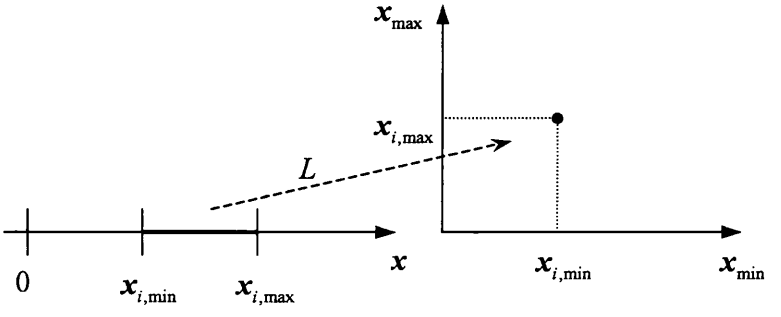


Figure 3.9 A segment in $\boldsymbol{R}^1$ space represented as a point in $\boldsymbol{R}^2$ space

Consequently a set of N geometric bodies in the n-dimensional Euclidean space $\boldsymbol{R}^n$ can be represented by a set of N points $P_i, i=1,N$ in 2n-dimesional space $\boldsymbol{R}^{2n}$.

3.2.7.2 Space bisection

Once the unique locations of bodies in 2n-dimensional space $\boldsymbol{R}^{2n}$ are defined, it is possible to create a data structure to store the information on the bodies with respect to their relative position in $\boldsymbol{R}^{2n}$ space. A spatial binary tree structure was developed by Bonet and Peraire [3.11], it is based on the principle of recursive bisection of the embedding space. Each node in the tree can have two children, left child and right child, and apart from the root node, also has a parent, i.e. there is one and only one

node pointing at it. A node without any child is defined as a *leaf*. Each node has a key field that decides which of its children will be accessed during the tree traversal for insertion of a node. For the spatial binary tree, each node is associated with a spatial sub-region in addition to the original data stored. A typical binary tree data structure is shown in Figure 3.10, and its corresponding storage allocation within computer memory is defined in Table 3.1.

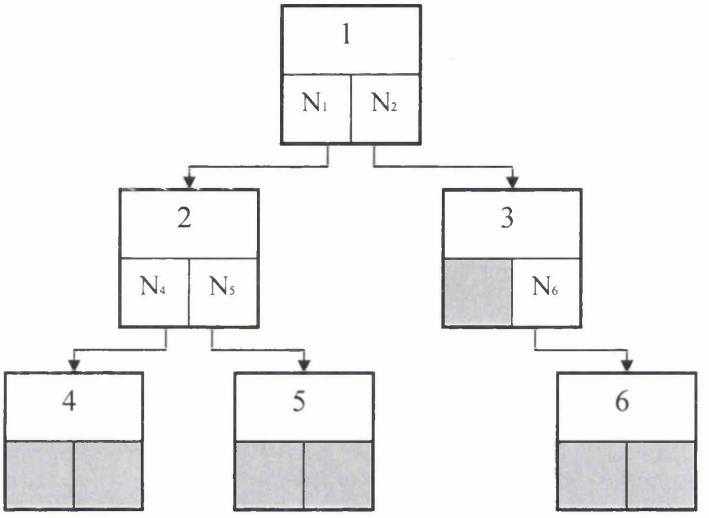


Figure 3.10 Representation of the binary tree data structure

Left child number	*Node number*	*Right child number*	*Parent Number*	J^{th}
2	1	3	0	1
4	2	5	1	2
0	3	6	1	2
0	4	0	2	1
0	5	0	2	1
0	6	0	3	1

Table 3.1 Binary tree data storage allocation

Table 3.1 denotes four integer arrays to be opened in data structure to store the lists of left and right child numbers, parent numbers, j^{th} direction, except for a list of node number, which are always listed sequentially by order.

A spatial binary tree represents the collection of N points, $\{\boldsymbol{P}_1, \boldsymbol{P}_2, \cdots, \boldsymbol{P}_N\}$, in a 2n-dimensinal space $\boldsymbol{R}^{2n}$. The generation of a spatial binary tree can start from the first point $\boldsymbol{P}_1$ as the root node, which represents the whole region $\boldsymbol{D}_0 = (\boldsymbol{c}_0, \boldsymbol{d}_0)$. $\boldsymbol{D}_0$ is the minimum bounding region of N points in $\boldsymbol{R}^{2n}$ space and is assigned to the root node in level 0. Through bisecting across the x^1 direction, $z_0^1 = \left(c_0^1 + d_0^1\right)/2$ is taken as a key for the root node. Second point $\boldsymbol{P}_2$ is inserted according to its lower corner coordinate $x_{2,\min}^1$, if $x_{2,\min}^1 < z_0^1$ then $\boldsymbol{P}_2$ is taken as a left child of the root and the left half region of $\boldsymbol{D}_0$, $c_0^1 \le x^1 \le z_0^1$, is assigned to node 2. If $x_{2,\min}^1 \ge z_0^1$ then point $\boldsymbol{P}_2$ is inserted as a right child and the right half region $z_0^1 \le x^1 \le d_0^1$ is assigned to the node 2. The sub-region $\boldsymbol{D}_1 = (\boldsymbol{c}_1, \boldsymbol{d}_1)$ is further bisected across the direction of x^2 and $z_1^2 = \left(c_1^2 + d_1^2\right)/2$ is taken as the key for nodes in level 1. The insertion of the third point is started from the root node and the direction x^1, locating its position in the two sub-region depending on the condition of $x_{3,\min}^1 < z_0^1$, then checks if the corresponding left or right child node is occupied. If it is free the third node is inserted into the tree, otherwise treat child node in level 1 as the root node and compare the lower corner coordinate $x_{3,\min}^2$ with z_1^2 and then repeat the same procedure described above. The tree is completed when N points in the $\boldsymbol{R}^{2n}$ space are inserted.

Generally if a node m is at the hierarchy level k of the binary tree, the sub-region of the node m is defined as $\boldsymbol{D}_k = (\boldsymbol{c}_k, \boldsymbol{d}_k)$. The half sub-regions associated with its left and right child are $\boldsymbol{D}_{kl} = (\boldsymbol{c}_{kl}, \boldsymbol{d}_{kl})$ and $\boldsymbol{D}_{kr} = (\boldsymbol{c}_{kr}, \boldsymbol{d}_{kr})$ respectively, resulting from the bisection of $\boldsymbol{D}_k$ by a plane normal to the j^{th} coordinate axis, where j is chosen cyclically from the n-dimensional space as

$$j = 1 + \mathrm{mod}(k, n) \tag{3.63}$$

Equation (3.63) is not explicitly calculated and j is updated during insertion of the nodes. The list of j^{th} direction of the nodes is stored in an integer array, shown in Table 3.1, for later contactor detection search. The key for the node m is set to be $z_k^j = \left(c_k^j + d_k^j\right)/2$, and the sub-regions $(\boldsymbol{c}_{kl}, \boldsymbol{d}_{kl})$, $(\boldsymbol{c}_{kr}, \boldsymbol{d}_{kr})$ are defined by

$$c_{kl}^i = c_k^i,\ d_{kl}^i = d_k^i \ \ for\ i \neq j\ \ c_{kl}^j = c_k^j,\ d_{kl}^j = z_k^j \tag{3.64}$$

$$c_{kr}^i = c_k^i,\ d_{kr}^i = d_k^i \ \ for\ i \neq j\ \ c_{kr}^j = z_k^j,\ d_{kr}^j = d_k^j \tag{3.65}$$

Remark:

(1) The shape of the tree obtained in above procedure depends on the spatial distribution of the N points and somewhat the order in which the points were inserted. For a well balanced tree, the cost of generating a binary tree is proportional to $N\log(N)$.

(2) The region associated with a given node m in the tree contains all the sub-regions associated with nodes descended from m and all points stored in these nodes lie inside the region. This is the most important feature of the spatial binary tree.

(3) In the ADT structure a node k represents a region $(\boldsymbol{c}_k, \boldsymbol{d}_k)$ and contains a point $\boldsymbol{P}_k = \left(\boldsymbol{x}_{k,\min}, \boldsymbol{x}_{k,\max}\right)$, which represents a rectangular target facet; the following condition must be satisfied.

$$\boldsymbol{c}_k \leq \boldsymbol{x}_{k,\min} \quad \boldsymbol{x}_{k,\max} \leq \boldsymbol{d}_k$$

3.2.7.3 Bounding box intersection search

A contactor detection search can be considered as a geometrical intersection problem. Assuming a contactor node, which is circumscribed with an axis aligned bounding box, and represented by a point $\boldsymbol{P} = (\boldsymbol{a}, \boldsymbol{b})$ in $\boldsymbol{R}^{2n}$ space. The bounding box of the

contactor node is extended by a buffer zone l_{buff} of contact detection; the larger the zone, the more expensive the contact interaction computations. Normally the buffer zone is taken as the maximum length of target facets. The intersection between the contactor node and the region represented by the node k, namely $(\boldsymbol{c}_k, \boldsymbol{d}_k)$, is checked by the following condition.

$$\boldsymbol{a} \le \boldsymbol{d}_k \qquad \boldsymbol{b} \ge \boldsymbol{c}_k$$

or

$$a^i \le d_k^i \qquad b^i \ge c_k^i \qquad i = 1, n \tag{3.66}$$

If condition (3.66) is not satisfied, then the complete set of point stored in the sub-tree rooted at k can be disregarded from the search, thus avoiding the need to examine the coordinate of every single point. If the region associated with node k in the tree intersects with the contactor range $(\boldsymbol{a}, \boldsymbol{b})$, then a geometric searching algorithm emerges in a recursive form as

(1) Check if the coordinates of the node k in the sub-tree root, $\boldsymbol{P}_k = (\boldsymbol{x}_{k,\min}, \boldsymbol{x}_{k,\max})$ intersect $(\boldsymbol{a}, \boldsymbol{b})$, i.e. check whether $a^i < x_{k,\max}^i$, $x_{k,\min}^i < b^i$ for $i = 1, n$. If the condition is satisfied, update number and list of possible contacted target facets for the contactor node.

(2) If the left child of the sub-root is non-zero and the region $(\boldsymbol{c}_{kl}, \boldsymbol{d}_{kl})$ intersects with $(\boldsymbol{a}, \boldsymbol{b})$, i.e. if $a^i \le d_{kl}^i$, $c_{kl}^i \le b^i$, $i = 1, n$ search the left sub-tree.

(3) If the right child of the sub-root is non-zero and the region $(\boldsymbol{c}_{kr}, \boldsymbol{d}_{kr})$ overlaps with $(\boldsymbol{a}, \boldsymbol{b})$, i.e. if $a^i \le d_{kr}^i$, $c_{kr}^i \le b^i$, $i = 1, n$ search the right sub-tree.

From the geometric searching procedure, a list of points, which represent target facets that intersect the contactor range $(\boldsymbol{a}, \boldsymbol{b})$, can be found for a contactor node. The searching procedure is carried out for all contactor nodes under consideration.

3.2.7.4 Local contact resolution

After global spatial search a short list of potential contact target facets are identified for each contactor node. The local contact resolution phase requires a detailed geometric description of the target facets in order to find out the actual intersected target facets. A simple criterion to detail possible intersected target facets in 2D and 3D contact cases is given by Hallquist *et al* [3.2].

3.3 Continuum adaptive remeshing

The key steps associated with a continuum adaptive remeshing algorithm are listed as follows, which we will discuss later in detail.

- Geometry entity related model definition
- Error estimation and prediction of mesh density
- Re-generation of the new mesh by an automatic mesh generator
- Field values mapping between the two meshes

3.3.1 Geometry entity related model definition

In adaptive analysis the configuration of a solid body is defined by a set of hierarchical database using geometry entities and operation assignments. For example in 3D case, a solid body may be composed of a geometry volume, which contains several geometry surfaces, lines and geometry vertices, these geometry entities are set by a graphical pre-processor. All geometry entities are related with mesh data, i.e. nodes, elements and element faces. A typical geometry data are shown in Table 3.2.

Volumes geometry_volume { volume number nodes { number of mesh nodes node number list } elements { number of elements element number list }
Surfaces geometry_surface { surface number nodes { number of mesh nodes node number list } elements { number of elements element number list } faces { number of element faces element face number list }
Lines geometry_line { line number nodes { number of mesh nodes node number list } elements { number of elements element number list } faces { number of element faces element face number list }
Points geometry_vertex { vertex number nodes { number of mesh nodes node number list }

Table 3.2 Geometry entities related nodes, elements and element faces

All the operating assignment data such as element topology, element geometry properties, global and element load, support data, element initial data, etc. are defined by geometry entities and its geometry number. Generally geometry entities are not changed throughout the simulation, only geometry related nodes, elements and element faces are changed after each mesh adaptation. But in some cases, the geometry entity may be deactivated and be removed due to their related element

erosion or boundary element surfaces sticking together. A new geometry entity may be created if the original geometry is split into two or more pieces, which often happens in fracture or rock blasting simulations. Therefore, the analysis program should be capable of handling the relevant geometry database intelligently.

3.3.2 Error estimation and prediction of mesh density

Error estimations based on the energy norm, velocity gradients and the plastic work rate are three most efficient and adequate error indicators for fluid and solid structure problems, the detailed description of the three error estimations are introduced as following:

Error estimate based on the energy norm

The energy norm in the linear model of the Eulerian and Lagrangian flow [3.14][3.22] can be defined as

$$\|u\|^2 = \int_\Omega \boldsymbol{s}^T (2\mu)^{-1} \boldsymbol{s} d\Omega + \int_\Omega P(\lambda + \frac{2}{3}\mu)^{-1} P d\Omega \qquad 3.67$$

and the error in the energy norm therefore is

$$\|e\|^2 = \sum_{k=1}^{M} \|e\|_k^2 = \int_\Omega (\boldsymbol{s} - \hat{\boldsymbol{s}})^T (2\mu)^{-1} (\boldsymbol{s} - \hat{\boldsymbol{s}}) d\Omega + \int_\Omega \left(P - \hat{P}\right)(\lambda + \frac{2}{3}\mu)^{-1} \left(P - \hat{P}\right) d\Omega \qquad 3.68$$

where $\|e\|_K$ is the k^{th} element error norm, and M is the number of elements in the domain. The element $\hat{\boldsymbol{s}}$ and $\hat{P}$ are finite element solutions for the deviatoric stress tensor and pressure respectively, $\boldsymbol{s}$ and P are the corresponding exact solutions. μ is the viscosity and λ is a volumetric viscosity coefficient analogous to the bulk modulus in linear elasticity.

Remark: Although the error estimate in the energy norm is not strictly applicable in the nonlinear problem, it is still a useful error indicator.

Error estimate based on the velocity gradients

The L_2-norm of the velocity gradient is defined as

$$\|u\|^2 = \int_\Omega (\nabla \boldsymbol{u})^T (\nabla \boldsymbol{u}) d\Omega \tag{3.69}$$

and the L_2-norm of the error in their representation is set by

$$\|e\|^2 = \sum_{k=1}^{M} \|e\|_k^2 = \int_\Omega (\nabla \boldsymbol{u} - \nabla \hat{\boldsymbol{u}})^T (\nabla \boldsymbol{u} - \nabla \hat{\boldsymbol{u}}) d\Omega \tag{3.70}$$

where $\nabla \hat{\boldsymbol{u}}$ is the discontinuous, approximated element velocity gradient, obtained from a finite element solution. $\nabla \boldsymbol{u}$ is the corresponding exact solution.

Error estimate based on the plastic work rate

The plastic work rate can be expressed as

$$\|u\|^2 = \int_\Omega \boldsymbol{\sigma}^T \dot{\boldsymbol{\varepsilon}}_p d\Omega \tag{3.71}$$

The error associated with the plastic dissipation is defined by

$$\|e\|^2 = \sum_{k=1}^{M} \|e\|_k^2 = \int_\Omega (\boldsymbol{\sigma} - \hat{\boldsymbol{\sigma}})^T \left(\dot{\boldsymbol{\varepsilon}}_p - \hat{\dot{\boldsymbol{\varepsilon}}}_p\right) d\Omega \tag{3.72}$$

where $\hat{\boldsymbol{\sigma}}$ and $\hat{\dot{\boldsymbol{\varepsilon}}}_p$ are finite element solutions for the Cauchy stress and plastic strain rate tensor respectively, $\boldsymbol{\sigma}$ and $\dot{\boldsymbol{\varepsilon}}_p$ are the corresponding exact Cauchy stress and plastic strain rate tensor.

As the exact solutions of $\boldsymbol{s}$, P, $\boldsymbol{\sigma}$, $\dot{\boldsymbol{\varepsilon}}_p$ and $\nabla \boldsymbol{u}$ in equations (3.67)-(3.72) are unknown, we can use some higher order approximations ${}^*\boldsymbol{s}$, ${}^*P$, ${}^*\boldsymbol{\sigma}$, ${}^*\dot{\boldsymbol{\varepsilon}}_p$ and ${}^*\nabla \boldsymbol{u}$ instead of the exact solutions. ${}^*\boldsymbol{s}$, ${}^*P$, ${}^*\boldsymbol{\sigma}$, ${}^*\dot{\boldsymbol{\varepsilon}}_p$ and ${}^*\nabla \boldsymbol{u}$ are obtained by least square smoothing [3.23] or some other projection method, e.g. nodal averaging method [3.24]. These procedures lead to higher order approximations, which tends to the exact solution. Therefore, the norm $\|u\|$ and error $\|e\|$ in equation (3.67)-(3.72) are replaced by $\|\hat{u}\|$ and $\|\hat{e}\|$.

A global relative error η for the three types of error estimate can be written as

$$\eta = \frac{\|\hat{e}\|}{\|\hat{u}\|} < \bar{\eta} \tag{3.73}$$

where $\bar{\eta}$ is a target allowable error value. The global relative error is used as a check for remeshing; if equation (3.73) is violated then the remeshing procedure is required. For transient fluid problems, this criterion should be satisfied at all times. When remeshing is required, the corresponding target error for a quality element needs to be defined as

$$\|\hat{e}\|_{k,target} = \bar{\eta} \frac{\|\hat{u}\|}{M^{\frac{1}{2}}} \tag{3.74}$$

Then the ratio of the element estimate error and element target error, ξ_k, can be set for each element as

$$\xi_k = \frac{\|\hat{e}\|_k}{\|\hat{e}\|_{k,target}} \qquad k = 1, M \tag{3.75}$$

ξ_k is also called the element error indicator or refinement indicator. When $\xi_k > 1$, a finer element mesh size is required, whereas the mesh size may be coarsened if $\xi_k < 1$. The required element size or new element mesh density can be predicted by

$$\bar{h}_k = \frac{h_k^{old}}{\xi_k} \qquad k = 1, M \tag{3.76}$$

where h_k^{old} and $\bar{h}_k$ are the current and predicted element sizes respectively. Equation (3.76) gives a discontinuous distribution of element sizes throughout the domain. However commercially used mesh generators normally require a continuous distribution of prescribed element sizes as input data to generate a new mesh. Therefore a smoothing procedure, either using least squares fitting or nodal averaging, projects the discontinuous element size distribution onto a continuous basis.

The implementation of the error estimate and mesh density prediction using a nodal averaging method is summarized in Figure 3.11

(1) Do $k=1,M$

Project $\hat{s}$, $\hat{P}$ or $\hat{\boldsymbol{\sigma}}, \hat{\dot{\boldsymbol{\varepsilon}}}_p$ onto element nodes

Enddo

Averaging $\hat{s}$, $\hat{P}$ or $\hat{\boldsymbol{\sigma}}, \hat{\dot{\boldsymbol{\varepsilon}}}_p$ get ${}^*s$, ${}^*P$ or ${}^*\boldsymbol{\sigma}, {}^*\dot{\boldsymbol{\varepsilon}}_p$

(2) Set $\|\hat{u}\|=0$, $\|\hat{e}\|=0$

Do $k=1,M$

Interpolate ${}^*s$, ${}^*P$ or ${}^*\boldsymbol{\sigma}, {}^*\dot{\boldsymbol{\varepsilon}}_p$ at Gauss point

Calculate $\|\hat{u}\|_k$, $\|\hat{e}\|_k$ using (3.67)(3.68) or (3.71)(3.72)

Update $\|\hat{u}\| = \|\hat{u}\| + \|\hat{u}\|_k$

$\|\hat{e}\| = \|\hat{e}\| + \|\hat{e}\|_k$

Enddo

(3) Calculate η using (3.73)

if $\eta < \bar{\eta}$ then

Exit

Elseif

(4) Set $\|\hat{e}\|_{k,target}$ using (3.74)

Do $k=1,M$

Define error indicator ξ_k using (3.75)

Mesh density $\bar{h}_k$ using (3.76)

Project $\bar{h}_k$ on element nodes

Enddo

Average $\bar{h}_k$ to obtain ${}^*\bar{h}_k$

Endif

Figure 3.11 Error estimate and mesh density prediction

From Figure 3.11 it is seen that the whole procedure of error estimation and mesh density prediction may involve loop over elements in the domain three times. It has to be mentioned that there are another two different error indicators to trigger the remeshing; one is the element Jacobian or volume based distortion error indicator, the other is an element nodal angle change based error indicator. The distortion error indicator based on the element Jacobian change can be defined as

$$\eta = \max_k \left(\xi_k\right) = \max_k \left(\frac{J_c^k - J_0^k}{J_0^k} \right) \quad k=1,2,\ldots M \tag{3.77}$$

where J_c^k and J_0^k are the k^{th} element deformed and initial Jacobian respectively. The element nodal angle change based distortion error indicator is defined as

$$\eta = \max_k(\xi_k) = \max_k\left(\frac{\alpha_{i,c}^k - \alpha_{i,0}^k}{\alpha_{i,0}^k}\right) \quad k = 1,2,\ldots M \quad i = 1, n_d \qquad 3.78$$

where $\alpha_{i,c}^k$ and $\alpha_{i,0}^k$ are the k^{th} element i^{th} corner nodal deformed and initial angles respectively. Equation (3.78) is only applicable for 2d cases.

3.3.3 Re-generation of the new mesh by automatic mesh generator

Automatic mesh generation has been the subject of much research in many applications of finite element analysis [3.24] [3.25]. The advancing front technique developed by Peraire *et al* [3.21] is a most efficient and popular method used to generate 2-D triangle or 3-D tetrahedral linear elements. The advancing front method, along with the later developed Delaunay method, belongs to the so-called '*h*' refinement process in which increased accuracy is achieved by variation of the element size, i.e. mesh density. In contrast to the '*h*' refinement process, '*p*' refinement changes the order of the element polynomial interpolation function, but it has a limited general applicability in practicable cases.

The basic idea behind the advancing front technique is that it first sets a background grid for the new mesh with its nodal mesh density predicted by the error estimator and defines a set of initial front facets on the boundary surface. Then it chooses a front facet as a base to generate a new node and a new element inside of the domain, according to the mesh density provided by the background grid. After new elements are generated on the boundaries, the front is advanced by shifting to appropriate new facets. By repeatedly generating new nodes, new elements and advancing to the new front, until a set of front facets is not found, the whole process is completed.

3.3.4 Field values mapping between the two meshes

When a new mesh is produced, mapping of nodal displacements, velocities, temperature and history-dependent variables at element Gauss points from the old mesh to the newly generated mesh is required. There are several important aspects that have to be met for a good mapping scheme, as addressed by Peric [3.17] and Lee *et al* [3.18] and listed as follows

1) Consistency with the constitutive equations
2) Requirement of equilibrium and minimisation of the numerical diffusion of the transferred state fields
3) Compatibility of the history-dependent internal variables transferred
4) Compatibility with evolving boundary conditions

In order to satisfy the above four requirements there are two kinds of mapping scheme used in commercial finite element analysis codes. One is the background element mapping and other is the weighted least squares mapping method. The background element mapping is a popular early method [3.15][3.16][3.17], which consists in extrapolating the history-dependent variables at element Gauss points to the element nodes of the old mesh, mapping nodal variables from old to new mesh and then performing element interpolation at the Gauss points of the new mesh. For any new point it is necessary to find a background element in the old mesh, which includes the point. Then it requires calculation of the local coordinates of this point in the background element and interpolation of the old nodal variables to the new point. A schematic of the mapping technique is given by [3.17], which will not be discussed here in detail. Since the background element mapping scheme involves Gauss point to nodes and nodes to Gauss point repeated interpolations it costs more computation time and smoothes the numerical solution, which may cause unnecessary numerical diffusion. Experience has found that simulations, which involve significant changes in geometry and sharp gradients of field values, can inherently suffer from excessive amounts of diffusion of field variables. Consequently, the weighted least squares mapping operator has received much research attention [3.27].

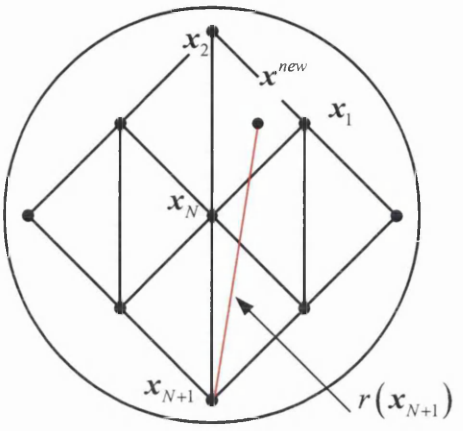


Figure 3.12 Least square mapping scheme

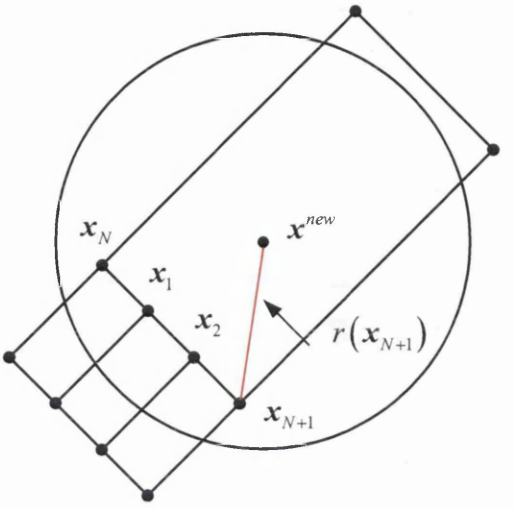


Figure 3.13 Singularity of matrix C_{jk}, $jk = 0, m$

Considering a newly generated point at x^{new} surrounded by $(N+1)$ neighbouring points $x_i, i = 1, N+1$, on the old mesh, as shown in Figure 3.12 and $y_i, i = 1, N$ represent the variables at x_i on the old mesh. The estimated values $\varphi_m(x)$ at any point x can be simply defined by a polynomial function

$$\begin{aligned}\varphi_m(x) &= a_0 P_0(x) + a_1 P_1(x) + \ldots a_m P_m(x) \\ &= \sum_{K=0}^{m} a_K P_K(x)\end{aligned} \tag{3.79}$$

where $P_K(x)$ is the k^{th} polynomial base for a weighted least square mapping. $a_0, a_1, \ldots a_m$ are the parameters to be determined. A weighted least square function $\psi(a_0, a_1, \ldots a_m)$ can be defined by

$$\begin{aligned}\psi(a_0, a_1, \ldots a_m) &= \sum_{i=1}^{N} w_i \left[\varphi_m(x_i) - y_i\right]^2 \\ &= \sum_{i=1}^{N} w_i \left[\sum_{j=0}^{m} a_j P_j(x_i) - y_i\right]^2\end{aligned} \tag{3.80}$$

where w_i is a weighting function, which is set by a cosine function,

$$w_i = \cos^2\left(\frac{\pi \left\| x_i - x^{new} \right\|}{2r(x_{N+1})}\right) \tag{3.81}$$

where N stands for the number of closest neighbouring points, which are used to map a new variables at $\boldsymbol{x}^{new}$. $\boldsymbol{x}_i, i = 1,2,\ldots N$ are the spatial positions of N closer neighbouring points. $\boldsymbol{x}_{N+1}$ is the spatial position of the $(N+1)^{th}$ closest neighbouring point and $r(\boldsymbol{x}_{N+1})$ is the distance from $\boldsymbol{x}^{new}$ to $\boldsymbol{x}_{N+1}$. Minimizing the function (3.80) leads to solution of the following equivalent system with unknown parameters $a_k, k = 0,1,\ldots m$

$$\frac{\partial \psi}{\partial a_K} = 0 \qquad k = 0,1,\ldots m \tag{3.82}$$

$$\sum_{i=1}^{N} w_i \left[\sum_{j=0}^{m} a_j P_j(\boldsymbol{x}_i) - y_i \right] P_K(\boldsymbol{x}_i) = 0 \tag{3.83}$$

Here we introduce the following notations

$$C_{jk} = \sum_{i=1}^{N} w_i P_j(\boldsymbol{x}_i) P_K(\boldsymbol{x}_i) \tag{3.84}$$

$$C_K = \sum_{i=1}^{N} w_i y_i P_K(\boldsymbol{x}_i) \tag{3.85}$$

Then equation (3.83) could be rewritten as

$$\sum_{j=0}^{m} C_{jk} a_j = C_K \qquad k = 0,1,\ldots m \tag{3.86}$$

By solving $(m+1)$ equations of (3.86), the newly interpolated value at $\boldsymbol{x}^{new}$ can be calculated by equation (3.79) as

$$y^{new} = \varphi_m(\boldsymbol{x}^{new}) = \sum_{K=0}^{m} a_K P_K(\boldsymbol{x}^{new}) \tag{3.87}$$

If a general interpolation function is defined by a linear polynomial set, then m = 3 and the dimension of $C_{jk}, j,k = 0,3$ is 4×4 for the 3-D case, and m = 2 with dimension of $C_{jk}, j,k = 0,2$ is 3×3 for 2-D case. Since the least squares mapping scheme does not need extrapolation of the history dependent variables from element Gauss points to the element nodes, it is more accurate than the background element

mapping. It has to be mentioned that a special case would occur when $(N+1)$ neighbouring points x_i, which are closer to the new point x^{new} cluster on a line, as shown in Figure 3.13, or on a plane for the 3-D case. From our experience it is often occurred in 3-D membrane and shell adaptive remeshing analysis. In this case the matrix C_{jk} is near singularity and the determinant of matrix C_{jk} tends to zero. In order to avoid those situations, a mapping tolerance value η_{tol} is introduced. When the determinant of matrix C_{jk} is less than η_{tol}, it transfers the global coordinates of $x_i, i=1, N+1$ and x^{new} into a local coordinate system, which is parallel to the line or the plane. The weighted least squares mapping is now carried out on the local coordinate system. The singularity of matrix C_{jk} is avoided by reducing its dimension by 1.

The following example defined in reference [3.28] illustrates the use of different mapping methods to transfer nodal variables and history-dependent state variables at the Gauss point. The problem undertaken is an elasto-plastic stress analysis of a billet which is being extruded through a tapered die with a wall angle of 20 degrees by a piston pushed at a velocity of 6000mm/s, as shown in Figure 3.14. The objective is to define the effective plastic strain distribution on the deformed billet.

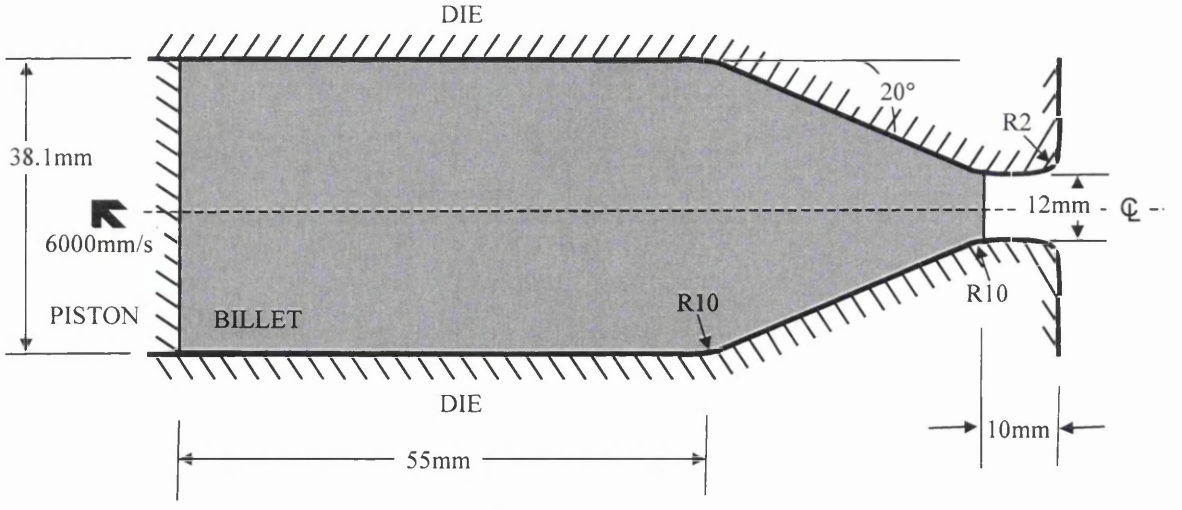


Figure 3.14 Billet extruded through a tapered die

The problem is axisymmetric about the horizontal axis and so only one-half of the billet/tool configurations will be modelled, with suitable constraints placed along the symmetric boundary. The tools are set to be rigid with the velocity prescribed on the piston as a whole and the die fully restrained, so plastic deformation can only occur in the billet. Unstructured linear quadrilateral 4 noded elements are used. Contact between the billet and tools is modelled using a slideline contact algorithm, taking into account the effects of friction. A friction coefficient of 0.2 is used in the analysis.

The billet is expected to undergo large deformation around the curved sections of the die and the finite element mesh will become excessively distorted if the same mesh is used continuously throughout the analysis. Extra mesh refinement is specified in these regions, and the billet is remeshed several times, transferring the displacement and state variables from the old to the new mesh. The mesh adaptation is checked every 500 iterations for an allowable distortion error of 5% and at each adaptation an error estimator based on the plastic work rate is used to control the size of the elements. The explicit formulation is used to perform this adaptive analysis.

The material for the billet is modelled using the Von-Mises isotropic plasticity model. The definition of the uniaxial yield stress, together with the hardening curve values are defined in Table 3.3 and Table 3.4.

Property	***Value***
Young's Modulus	6.896×10^4 N/mm^2
Poisson's Ratio	0.32
Density	2.8×10^{-8} Ns^2/mm^4
Uniaxial Yield Stress	31 N/mm^2

Table 3.3 Material Properties for Billet

Effective Plastic Strain	***Effective Stress (N/mm^2)***
0	31
10	2643

Table 3.4 Material Hardening Curve for Billet

The piston and die materials are modelled as elastic solids with the properties listed in Table 3.5.

Property	*Value*
Young's Modulus	2.1×10^5 N/mm^2
Poisson's Ratio	0.3
Density	7.8×10^{-6} Ns^2/mm^4

Table 3.5 Material Properties for Tools (Die and Piston)

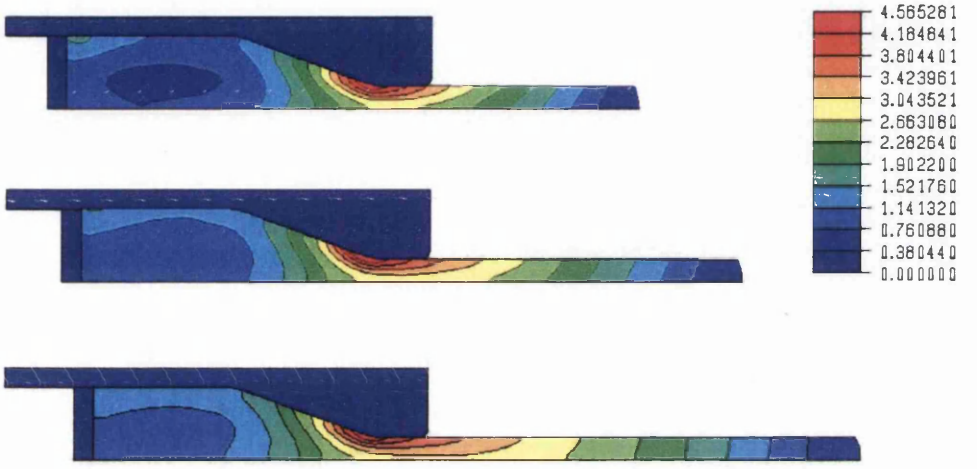


Figure 3.15 Sequence of effective plastic strains using Background mapping scheme

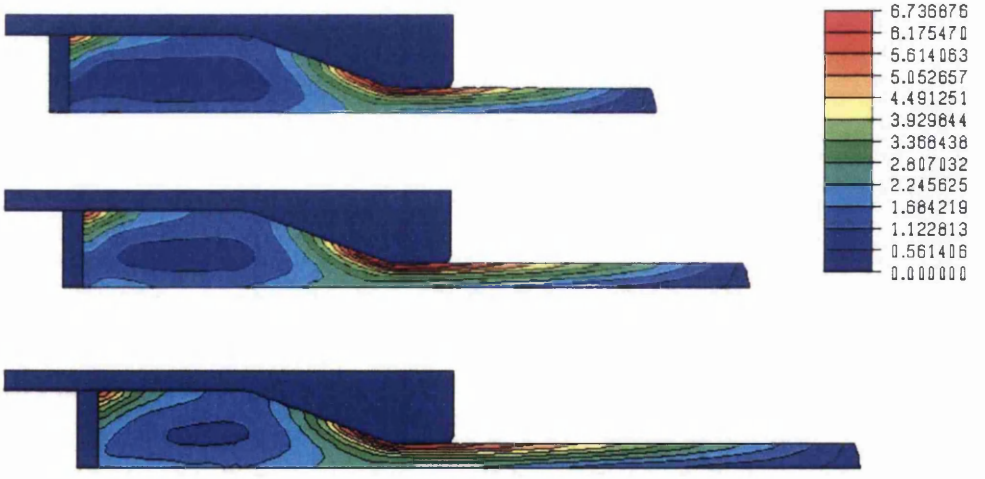


Figure 3.16 Sequence of effective plastic strains using Weighted least-square mapping scheme

Both the background element mapping and the weighted least square mapping are used to transfer the nodal displacement and stresses, strains, plastic strains and effective plastic strains at Gauss points. Figure 3.15 and Figure 3.16 shows the sequence of fixed contour of effective plastic strain at time 0.00179, 0.00239 and 0.00298 second. It is obvious that the weighted least square method gives a more realistic effective plastic strain distribution than the one obtained from the background element mapping. The layered plastic strain from the tapered die maintains a layered contour profile by the weighted least mapping, but for the background element mapping the pattern of layered plastic strain contour is gradually smoothed out.

3.4 References

[3.1] J.O. Hallquist, NIKE2D, An implicit, finite deformation, finite element code for analysing the static and dynamic response of two-dimensional solids, *Rept. UCRL-52678*, University of California, Lawrence Livermore National Laboratory, 1979.

[3.2] J.O. Hallquist, G.L. Goudreau and D.J. Benson, Sliding interfaces with contact-impact in large-scale Lagrangian computations, *Comp. Meth. Appl. Mech. Engng.* 51 (1985) 107-137.

[3.3] P. Wriggers and J.C. Simo, A note on the tangent stiffness for fully nonlinear contact problems, *Commun. Appl. Numer. Meth.* 1 (1985) 199-203.

[3.4] D. Peric and D.R.J. Owen, Computational model for 3-D contact problems with friction based on the penalty method, *Int. J. Num. Meth. Engng.* Vol.35 (1992) 1289-1309.

[3.5] T. Belytschko and M.O. Neal, Contact-impact by the pinball algorithm with penalty and Lagrangian methods, *Int. J. for Num. Engng.,* Vol.31 (1991) 547-572.

[3.6] J.C. Simo and T.A. Laursen, An augmented Lagrangian treatment of contact problems involving friction, *Comp. Struct.* 42 (1992) 97-116.

[3.7] M.A. Crisfield, Non-linear finite element analysis of solids and structures: Volume 2 advanced topics. (Wiley, 1997) 411-446.

[3.8] T.A. Laursen and J.C. Simo, A continuum-based finite element formulation for the implicit solution of multi-body, large deformation frictional contact problems, *Int. J. Num. Meth. Eng.* 36 (1993) 3451-3485.

[3.9] E.A. de Souza, K. Hashimoto, D. Peric and D.R.J. Owen, A phenomenological model for frictional contact of coated steel sheets, *In Proceedings, NUMISHEET*, Tokyo, Japan,1993.

[3.10] K. Hashimoto, E.A. de Souza Neto, D. Peric and D.R.J. Owen, A study on dynamic frictional behaviour of coated steel sheets: experiment, formulation and finite element simulations, *Report of INME*, University College of Swansea.

[3.11] J. Bonet and J. Peraire, An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems, *Int. J. Numer. Meth. Engng.* 31 (1991) 1-17.

[3.12] Y.T. Feng and D.R.J. Owen, An augmented spatial digital tree algorithm for contact detection in computational mechanics, *Int. J. Numer. Meth. Engng.* 55 (2002) 159-176.

[3.13] I. Babuŝka and W.C. Rheinboldt, A-posteriori error estimates for finite element method. *Int. J. Numer. Meth. Engng.* 12 (1978) 1597-1615.

[3.14] R.E. Bank and A. Weiser, Some a posteriori error estimators for elliptic partial differential equations, *Math. Comput.* 44 (1985) 283-301.

[3.15] O.C. Zienkiewicz and J.Z.Zhu, A simple error estimator and adaptive procedure for practical engineering analysis, *Int. J. Num. Meth. Engng.* 24, (1987) 337-357.

[3.16] O.C. Zienkiewicz and J.Z.Zhu, Superconvergent derivative techniques and a posteriori error estimation in the finite element method, Part 1: A general superconvergent recovery technique; Part 2 The Zienkiewicz-Zhu error estimator *Report of INME*, University College of Swansea, CR/671/91 and CR/672/91, 1991

[3.17] D. Peric, Ch. Hochard, M. Dutko and D.R.J Owen, Transfer operators for evolving meshes in small strain elasto-plasticity. *Comp. Meth. Appl. Mech. Engng.* 137 (1996) 331-344.

[3.18] N.S. Lee and K.J. Bathe, Error indicators and adaptive remeshing in large deformation finite element analysis, *Fin. Elem. Anal. Des.* 16 (1994) 99-139.

[3.19] J.T Oden, L. Demkowicz, W. Rachowicz and T.A. Westermann, Toward a universal h-p adaptive finite element strategy. Part 2: A posteriori error estimation, *Comp. Meth. Appl. Mech. Engng.* 77 (1989) 113-180.

[3.20] D. Peric, J. Yu and D.R.J. Owen, On error estimates and adaptivity in elasto-plastic solids: applications to the numerical simulation of strain localisation in classical and Cosserat continua, *Int. J. Num. Meth. Engng.* 37 (1994) 1351-1379.

[3.21] J. Peraire, M. Vahdati, K. Morgan and O.C. Zienkiewicz, Adaptive remeshing for compressible flow computations, *J. Comp. Phys.* Vol 72 (1987) 449-466.

[3.22] J. Wu, J.Z. Zhu, J. Szmelter and O.C. Zienkiewicz, Error estimation and adaptivity in Navier-Stokes incompressible flow, *Report of INME*, University College of Swansea, CR/647/90, 1990.

[3.23] E. Hinton and J. Campbell, Local and global smoothing of discontinuous finite element functions using least square methods, *Int. J. Num. Meth. Engrg.* 8 (1974) 461-480.

[3.24] M. Ainsworth, J.Z. Zhu, A.W. Craig and O.C. Zienkiewicz, Analysis of the Zienkiewicz-Zhu a posteriori error estimator in the finite element method, *Int. J. Num. Meth. Engrg.* 28 (1989) 2161-2174.

[3.25] S.H. Lo, A new mesh generation scheme for arbitrary planar domains, *Int. J. Num. Meth. Engrg.* 21. (1985) 1403-1426.

[3.26] W.J. Schroeder and M.S. Shepherd, A Combined Octree/Delaunay method for fully automatic 3-D mesh generation, *Int. J. Num. Meth. Engrg.* 29 (1990) 37-56.

[3.27] F. Olmi, E. Bittencourt and G.J. Creus, An interactive remeshing technique applied to two dimensional problems involving large ealsto-plastic deformations, *CIMNE*, Barcelona, 1997.

[3.28] O.C. Zienkiewicz, G.C. Huang and Y.C. Liu, Adaptive FEM computations of forming processes-application to porous and non-porous materials, *Int. J. Num. Meth. Engrg.* 30 (1990) 1527-1553.

Chapter 4

Implicit and explicit parallel solver

4.1 Introduction

Parallel computational strategies to perform numerical simulations have been extensively studied in the last two decades. A number of methods based on domain decomposition procedures have been proposed in recent years for parallel solution of both static and dynamic finite element equations of equilibrium. Among them the most popular methods are derived from sub-structuring techniques [4.1][4.2][4.3]. Typically, a large-scale finite element domain is decomposed into a set of subdomains and each of them is assigned to an individual processor. The solution of the local problem is naturally parallelized and a direct solution method is preferred for solving the sub-structure problem. With a little modification a sequential code can be directly used for this purpose. The resulting interface equations can be solved by both direct and iterative methods. The parallel implementations of direct or iterative solution of the resulting interface equations, that introduces a second level concurrency, have also been reported in the literature [4.4][4.5]. In section 4.2.1 we will introduce this technique in detail. The implementation of an implicit parallel solver, which is named the hybrid iterative direct parallel solver, is based on the non-overlapping domain decomposition and sub-structuring approach. It has to be mentioned that a method called finite element tearing and interconnecting parallel solution algorithm (FETI) [4.6][4.7] departs from classical substructure methods and offers an alternative way for the parallel finite element solution of equations, where a spatial domain is partitioned into a set of totally disconnected subdomains and Lagrange multipliers are introduced to enforce compatibility at interface nodes. In the static case, each floating

subdomain may induce a local singularity, which is overcome by eliminating the rigid body modes in each subdomain in parallel and these modes are related to the Lagrange multipliers through an orthogonal condition. Finally it solves the coupled equation system of local rigid modes and Lagrange multipliers by using a parallel conjugate projected gradient algorithm. This algorithm requires less inter-processor communications than substructure methods.

As we remark in section 4.2.2 the non-overlapping domain decomposition method may not be efficient for the problem that involves contact elements, either in the slide-line contact or discrete element contact approach. In such cases an overlapping domain decomposition method may be necessary. The parallelization of explicit finite element fluid dynamics with contact conditions is based on overlapping domain decomposition and the Schwarz alternating procedure. The implementation is relatively straightforward and a lot of research work [4.8][4.9][4.10] has been reported in this field. Because of the dual nature of the overlapping partitioning it means that communication requirements and cost may be slightly more than for the non-overlapping domain decomposition. But it offers a more efficient and flexible way to deal with contact interfaces, especially in combined finite-discrete element simulations [4.11].

The proposed implicit and explicit parallel solvers are implemented on a distributed memory computing system, which allows a large number of processors to be connected together with a high-bandwidth communication network. Distributed memory parallel environments, such IBM SP2, Intel Paragon, and the rapidly developed PC clusters in recent years, have the potential to provide the high capacity and computational speed necessary to make numerical simulation of large finite element analysis systems practical. The distributed memory environments follow a multiple-instruction/multiple-data (MIMD) paradigm, which allows for more flexibility in algorithm and program design. In this research work, parallelization is based on a commercial sequential code ELFEN, which is developed by Rockfield Software Ltd, with necessary modification. The message-passing between processors for data communications relies on the MPI library [4.12][4.13].

The outline of this chapter is as follows. Firstly domain decomposition methods are reviewed in section 4.2. Section 4.3 discusses the direct solution of linear systems, in which the modified Cholesky factorization is chosen for substructure condensation in the parallel implicit solver. Section 4.4 introduces iterative solution of linear systems and highlights a number of the iterative methods built on the Krylov subspace in detail. Implementation of the implicit parallel solver is described in section 4.5. In section 4.6 the parallelization of explicit finite element fluid dynamics is presented. Numerical examples to illustrate the parallel performance achieved will be presented in chapter 5.

4.2 Domain decomposition

The efficient use of a parallel computer requires two objectives to be achieved. First each processor must be kept busy doing useful work. Secondly, the amount of inter-processor communication must be kept small. The domain decomposition approach, which attempts to distribute computational work by breaking a large problem into a number of smaller subproblems, is undoubtedly the best known and perhaps the most promising technique to achieve these objectives. For many finite element computational problems, parallel processing can be achieved by dividing the whole problem domain into several subdomains according to the available processors. It involves distributing the subdomain data to each processor initially. A well-balanced situation can be achieved if each processor is assigned an equal number of elements. Each processor performs the same basic algorithm in concurrency on the subdomain and only communicates interface terms. If the interfacial nodes or elements that are shared by more than one subdomain are kept small, the inter-processor communications can be reduced to a minimum.

There are numerous techniques for domain decomposition, however they can be basically catalogued into two types:

Non-overlapping Domain Decomposition – It is also known as sub-structuring or Schur complement method; the finite element mesh is divided through the element

edges or faces for the 3-D case. Elements are assigned uniquely to subdomains, as shown in Figure 4.1. The nodes through which the cut is made are shared by adjacent subdomains and called 'interfacial' nodes, which are on the boundary $\Gamma_{q,p}$ and $\Gamma_{q,p} = \Gamma_{p,q}$ for the non-overlapping decomposition. The other nodes are 'internal' to the subdomain, so the equations are solved at those internal nodes without change. However at interfacial nodes it is necessary to assemble contributions from two or more subdomains. The non-overlapping domain decomposition method is adopted for implementation of an implicit parallel solver in this work.

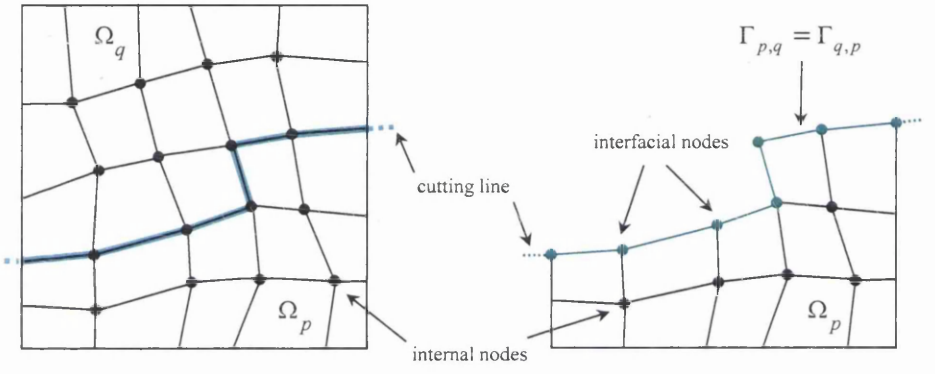


Figure 4.1 Non-overlapping Domain Decomposition

Overlapping Domain Decomposition – It is often known as Schwarz alternating procedure, originally developed by Schwarz (1869) to solve classical boundary value problems for linear elliptic equations. The partitioning cut is made across element edges, or faces for the 3-D case, as shown in Figure 4.2. The elements, which have been cut, are duplicated for both subdomain Ω_p and Ω_q adjacent to the cut. The area of cut elements below the cut is called the 'interfacial' boundary $\Gamma_{q,p}$ and the area of cut elements above the cut is called the 'external' boundary $\Gamma_{p,q}$ for subdomain Ω_p. The nodes on the boundary $\Gamma_{q,p}$ are called 'interfacial' nodes and on $\Gamma_{p,q}$ are called 'external' nodes. It should be noted that a partition gets to work not only with its interfacial nodes, but also with external nodes incident to the shared elements, which are 'owned' by other subdomains. Obviously, the number of boundary nodes that

include interfacial and external nodes is about twice that of non-overlapping partitioning.

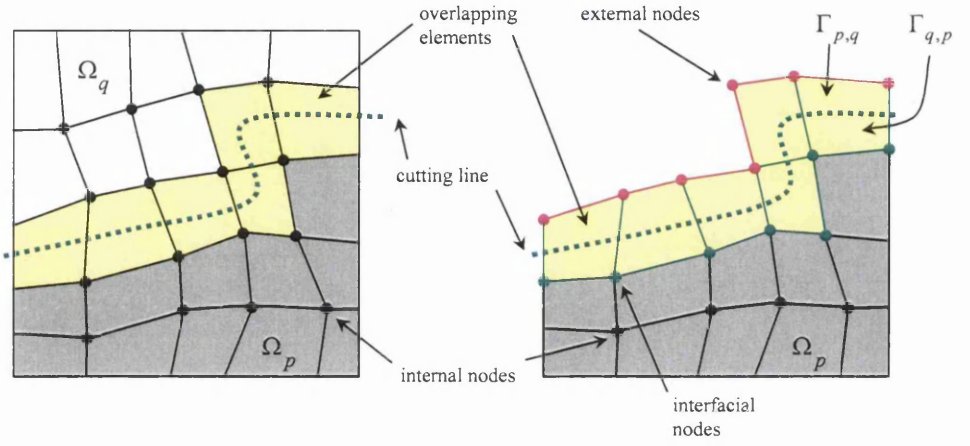


Figure 4.2 Overlapping Domain Decomposition

4.2.1 Non-overlapping Domain Decomposition

In the non-overlapping domain decomposition, the whole domain consists of mutually non-overlapping subdomains as

$$\Omega(t) = \bigcup_{p=1,s} \Omega_p(t) \qquad \Omega_p(t) \bigcap \Omega_q(t) = 0 \tag{4.1}$$

where s denotes the number of subdomains according to the available processors on a parallel computing system. A subdomain $\Omega_p(t)$ may change with time t for transient Stokes flow problems, which were already discussed in Chapter 2. In general, the resulting system of un-symmetric equations can be represented in block matrix notation as

$$\begin{bmatrix} \boldsymbol{K}_{11} & & & & \boldsymbol{K}_{1b} \\ & \boldsymbol{K}_{22} & & & \boldsymbol{K}_{2b} \\ & & \ddots & & \vdots \\ & & & \boldsymbol{K}_{ss} & \boldsymbol{K}_{sb} \\ \boldsymbol{K}_{b1} & \boldsymbol{K}_{b2} & \cdots & \boldsymbol{K}_{bs} & \boldsymbol{K}_{bb} \end{bmatrix} \begin{Bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_s \\ \boldsymbol{x}_b \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_s \\ \boldsymbol{b}_b \end{Bmatrix} \tag{4.2}$$

where each $\boldsymbol{x}_i$ represents an unknown variable vector defined at internal nodes of the subdomain Ω_p. The vector $\boldsymbol{x}_b$ represents an unknown variable vector defined at the interfacial nodes which are assembled by all subdomains and labelled last. The term $\boldsymbol{K}_{ii}$ represents a stiffness matrix corresponding to the unknown variables at internal nodes of the subdomain Ω_p. $\boldsymbol{K}_{bi}$ and $\boldsymbol{K}_{ib}$ correspond to the coupling terms between interfacial nodes and the internal nodes of subdomain Ω_p, for a symmetric system $\boldsymbol{K}_{ib} = \boldsymbol{K}_{bi}^T$. The term $\boldsymbol{K}_{bb}$ is a stiffness matrix related with all unknown variables at interfacial nodes. It should be mentioned that each of these matrices has been assembled from element stiffness matrices. Assuming that the assembly is considered only with respect to the subdomain Ω_p, an assembled local system of equations can be written as

$$\begin{bmatrix} \boldsymbol{K}_{ii} & \boldsymbol{K}_{ib} \\ \boldsymbol{K}_{bi} & \boldsymbol{K}_{bb}^p \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_i \\ \boldsymbol{x}_b^p \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_i \\ \boldsymbol{b}_b^p \end{bmatrix} \tag{4.3}$$

where $\boldsymbol{K}_{bb}^p$ contains only contribution from local elements that are in the subdomain Ω_p. The stiffness matrix $\boldsymbol{K}_{bb}$ and load vector $\boldsymbol{b}_b$ are the sum of $\boldsymbol{K}_{bb}^p$ and $\boldsymbol{b}_b^p$

$$\boldsymbol{K}_{bb} = \sum_{p=1}^{s} \boldsymbol{K}_{bb}^p \quad \boldsymbol{b}_b = \sum_{p=1}^{s} \boldsymbol{b}_b^p \tag{4.4}$$

By derivation from equation (4.2)-(4.4), a global Schur complement matrix, which is related with interfacial unknown variables, is defined as

$$\boldsymbol{S} = \sum_{p=1}^{s} \left[\boldsymbol{K}_{bb}^p - \boldsymbol{K}_{bi} \boldsymbol{K}_{ii}^{-1} \boldsymbol{K}_{ib} \right] = \sum_{p=1}^{s} \boldsymbol{S}_p \tag{4.5}$$

where $\boldsymbol{S}_p$ denotes the local Schur complement matrix. Equation (4.5) indicates that the global Schur complement can be easily obtained from local Schur complement matrices, and similarly for the load vector $\boldsymbol{y}$

$$\boldsymbol{y} = \sum_{p=1}^{s} \left[\boldsymbol{b}_b^p - \boldsymbol{K}_{bi} \boldsymbol{K}_{ii}^{-1} \boldsymbol{b}_i \right] = \sum_{p=1}^{s} \boldsymbol{y}_p \tag{4.6}$$

The forming of the local Schur complement $\boldsymbol{S}_p$ and local effective load vector $\boldsymbol{y}_p$ involves elimination of the internal unknown variables $\boldsymbol{x}_i$ and factorisation of matrix $\boldsymbol{K}_{ii}$. These procedures can be naturally operated on each processor in parallel, then $\boldsymbol{S}_p$ and $\boldsymbol{y}_p$ will be passed to the master processor. Assembly of the global Schur complement and load vector is carried out on the master processor.

The final condensed equations for solving the unknown variables at interfacial nodes are set as

$$\boldsymbol{S}\boldsymbol{x}_b = \boldsymbol{y} \tag{4.7}$$

In general, the global Schur complement tends to be smaller than the original $\boldsymbol{K}$ matrix, but also denser. The Schur complement matrix has an important property that if the original $\boldsymbol{K}$ is symmetric positive definite, then the $\boldsymbol{S}$ matrix is also symmetric positive definite. The direct solution of the reduced system (4.7) is the dominant part of the total computational time, and therefore an attractive alternative is to use an iterative solver, which will be discussed in detail in section 4.4. The implementation of an implicit parallel solver using hybrid iterative direct technique is introduced in section 4.5.

4.2.2 Overlapping Domain Decomposition

In the overlapping domain decomposition, subdomains are allowed to *overlap* each other such that

$$\Omega(t) = \bigcup_{p=1,s} \Omega_p(t) \qquad \Omega_p(t) \bigcap \Omega_q(t) \neq 0 \tag{4.8}$$

The resulting system of un-symmetric equations for the overlapping domain decomposition can be written in block matrix notation, using a reordering in which interfacial unknown variables are listed last in each subdomain

$$\begin{bmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{12} & \cdots & \boldsymbol{K}_{1s} \\ \boldsymbol{K}_{21} & \boldsymbol{K}_{22} & \cdots & \boldsymbol{K}_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{K}_{s1} & \boldsymbol{K}_{s2} & \cdots & \boldsymbol{K}_{ss} \end{bmatrix} \begin{Bmatrix} \hat{\boldsymbol{x}}_1 \\ \hat{\boldsymbol{x}}_2 \\ \vdots \\ \hat{\boldsymbol{x}}_s \end{Bmatrix} = \begin{Bmatrix} \hat{\boldsymbol{b}}_1 \\ \hat{\boldsymbol{b}}_2 \\ \vdots \\ \hat{\boldsymbol{b}}_s \end{Bmatrix} \tag{4.9}$$

In the p^{th} subdomain Ω_p, the unknown variable vector $\hat{\boldsymbol{x}}_p$ and load vector $\hat{\boldsymbol{b}}_p$ have the forms

$$\hat{\boldsymbol{x}}_p = \begin{Bmatrix} \boldsymbol{x}_{pI} \\ \boldsymbol{x}_{pb} \end{Bmatrix} \quad \hat{\boldsymbol{b}}_p = \begin{Bmatrix} \boldsymbol{b}_{pI} \\ \boldsymbol{b}_{pb} \end{Bmatrix} \tag{4.10}$$

where $\boldsymbol{x}_{pI}$ denotes an unknown variable sub-vector at internal nodes and $\boldsymbol{x}_{pb}$ represents an unknown variable sub-vector defined at the interfacial nodes of the subdomain p, as shown in Figure 4.2. The stiffness matrix $\boldsymbol{K}_{pp}$ can be split into four divisions, which are written in block matrix notation as

$$\boldsymbol{K}_{pp} = \begin{bmatrix} \boldsymbol{B}_p & \boldsymbol{E}_p \\ \boldsymbol{F}_p & \boldsymbol{C}_p \end{bmatrix} \tag{4.11}$$

In which $\boldsymbol{B}_p$ represents the stiffness sub-matrix associated with the internal variables and $\boldsymbol{C}_p$ is associated with the interfacial variables of subdomain p. $\boldsymbol{E}_p$ and $\boldsymbol{F}_p$ represent the coupling terms related with internal and interfacial nodes within the subdomain. The off-diagonal matrices $\boldsymbol{K}_{pq}$ and $\boldsymbol{K}_{qp}$ in equation (4.9) have the following forms

$$\boldsymbol{K}_{pq} = \begin{pmatrix} \boldsymbol{0}, & \boldsymbol{E}_{pq} \end{pmatrix} \qquad \boldsymbol{K}_{qp} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{E}_{qp} \end{pmatrix} \quad \forall p \neq q,\ p,q = 1,s \tag{4.12}$$

where $\boldsymbol{E}_{pq}$ and $\boldsymbol{E}_{qp}$ represent the coupling terms of the interfacial and external nodes shared by subdomain p and q, the majority of the elements of $\boldsymbol{E}_{pq}$ and $\boldsymbol{E}_{qp}$ are zero. A local system of equations for the subdomain p can be given by

$$\boldsymbol{B}_p \boldsymbol{x}_{pI} + \boldsymbol{E}_p \boldsymbol{x}_{pb} = \boldsymbol{b}_{pI} \tag{4.13a}$$

$$\boldsymbol{F}_p \boldsymbol{x}_{pI} + \boldsymbol{C}_p \boldsymbol{x}_{pb} + \sum_{q \in N_p} \boldsymbol{E}_{pq} \boldsymbol{x}_{qb} = \boldsymbol{b}_{pb} \tag{4.13b}$$

where $\boldsymbol{E}_{pq}\boldsymbol{x}_{qb}$ represent the forces contributed by the external nodes of the neighbouring subdomain q, the term N_p denotes a set of subdomains which are adjacent to subdomain p. If $\boldsymbol{B}_p$ is a non-singular matrix, and the unknown variable vector $\boldsymbol{x}_{pI}$ can be eliminated from equation (4.13), then

$$\boldsymbol{S}_p \boldsymbol{x}_{pb} + \sum_{q \in N_p} \boldsymbol{E}_{pq} \boldsymbol{x}_{qb} = \bar{\boldsymbol{b}}_{pb} \qquad p = 1, 2, \ldots s \tag{4.14a}$$

and

$$\boldsymbol{S}_p = \boldsymbol{C}_p - \boldsymbol{F}_p \boldsymbol{B}_p^{-1} \boldsymbol{E}_p \tag{4.14b}$$

$$\bar{\boldsymbol{b}}_{pb} = \boldsymbol{b}_{pb} - \boldsymbol{F}_p \boldsymbol{B}_p^{-1} \boldsymbol{b}_{pI} \tag{4.14c}$$

Finally, the system of equations (4.14) is to be solved for unknown variables $\boldsymbol{x}_{pb}$ at the interfacial nodes for each subdomain p, which involves unknown variables $\boldsymbol{x}_{qb,q \in N_p}$ at external nodes of the adjacent subdomains. Equations (4.14) are fully coupled. The global Schur complement matrix $\boldsymbol{S}$, which can be assembled from the local Schur complement matrices $\boldsymbol{S}_p$ and coupling terms $\boldsymbol{E}_{pq}$, is associated with the interfacial nodes of all subdomains and is given by

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{S}_1 & \boldsymbol{E}_{12} & \cdots & \boldsymbol{E}_{1s} \\ \boldsymbol{E}_{21} & \boldsymbol{S}_2 & \cdots & \boldsymbol{E}_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{E}_{s1} & \boldsymbol{E}_{s2} & \cdots & \boldsymbol{S}_s \end{bmatrix} \tag{4.15}$$

Obviously, the dimension of Schur matrix $\boldsymbol{S}$ in equation (4.15) is much larger than the one produced by the non-overlapping domain decomposition method. There are two ways for solving equations (4.14). One is using the same method as for non-overlapping domain decomposition, forming the local Schur matrices $\boldsymbol{S}_p$, coupling terms $\boldsymbol{E}_{qp,q \in N_p}$ and load $\bar{\boldsymbol{b}}_{pb}$ and passing them to the master processor, assembling the

global Schur matrix and load vector and solving the unknown variables of the interfacial nodes at the master processor. The other way is to solve equations (4.13) instead of (4.14) using the Schwarz alternating procedure, which includes the Multiplicative Schwarz procedure and Additive Schwarz procedure [4.21]. The basic steps of the Schwarz alternating procedure can be described as follow;

Step (1) Choose initial guess values for the interfacial unknown variables $\boldsymbol{x}_{pb}, p = 1, s$ of all subdomains.

Step (2) Until convergence is reached, loop over each subdomain.

Step (3) Solving equations (4.13) using the external variables $\boldsymbol{x}_{qb}, q \in N_p$, with initial guess values at first looping or updated new values at later looping, means that the external variables on the boundaries $\Gamma_{p,q}, q \in N_p$ of the adjacent subdomains are treaded as prescribed boundary conditions.

Step (4) Update interfacial variables x_{pb} on the boundaries $\Gamma_{q,p}, p \in N_q, q = 1, s$, which will be used as prescribed boundary conditions for the adjacent subdomains, and go back to step (2).

Remark;

(1) Since the dimension of the Schur matrix $\boldsymbol{S}$ in equation (4.15) is much larger than the one produced by the non-overlapping domain decomposition method, it is not necessary using the overlapping domain decomposition method to parallelize the implicit solver, if no contact situation is involved.

(2) If a numerical simulation involves contact problems, either by using slide-line contact or discrete element contact formulations, the non-overlapping domain decomposition method may not be efficient, since contact elements, which are generated internally and associated with the interfacial nodes, cannot be uniquely assigned to the subdomain. In this case the overlapping domain decomposition method may have to be used.

(3) In theory, the Schwarz alternating procedure is an iterative method, the Multiplicative Schwarz procedure is very reminiscent of block Gauss-Seidel iteration and the Additive Schwarz procedure is analogous to block Jacobi iteration. Obviously, the convergence problem will arise for solving equation (4.13), but the Schwarz alternating procedure opens a new way to parallelize

the explicit dynamic solver. Currently, parallelisation of explicit dynamic finite element analysis codes [4.8][4.9] is based on the Schwarz alternating procedure. The implementation of an explicit dynamic parallel solver is given in section 4.6.

4.3 Direct solution of linear systems

A system of linear equations is represented in the form

$$\boldsymbol{K}\boldsymbol{x} = \boldsymbol{b} \tag{4.16}$$

where $\boldsymbol{K}$ is a non-singular coefficient matrix $\boldsymbol{K} \in \boldsymbol{R}^{n \times n}$, $\boldsymbol{b}$ is the known right-hand side load vector $\boldsymbol{b} \in \boldsymbol{R}^n$, find solution vector $\boldsymbol{x} \in \boldsymbol{R}^n$ to satisfy equation (4.16). Many scientific problems lead to the requirement to solve linear systems of equations as part of the computations. The direct method theoretically gives the exact solution of a linear system within a predictable finite number of operations. The most obvious way of solving (4.16) is to find an inverse matrix of $\boldsymbol{K}$ and to apply the trivial multiplication with the load vector $\boldsymbol{b}$ as $\boldsymbol{x} = \boldsymbol{K}^{-1}\boldsymbol{b}$. However, this is an inefficient way of solving the equations since matrix inversion requires far too much computational resources. In addition, the $\boldsymbol{K}$ matrix is often well banded, while its inverse matrix is fully populated and demands much larger storage than that required for $\boldsymbol{K}$. Most direct methods optimise the process by transforming the coefficient matrix $\boldsymbol{K}$ into triangular or diagonal form in order to decrease the coupling between the equations. The commonly used direct methods are Gauss elimination, standard and modified Cholesky decomposition methods, specially the modified Cholesky factorisation with a banded, skyline profile scheme.

4.3.1 Gaussian elimination

Gauss elimination method is one of the most popular and numerically efficient approaches. The process can be carried out in $O(n^3)$ basic floating-point operations

with additions and multiplications. Many applications lead to linear systems with large n and it became soon evident that one has to exploit specific properties of $\boldsymbol{K}$ in order to make solution of the system feasible. This has led to variants of Gauss elimination in which the nonzero structure of $\boldsymbol{K}$ is exploited, so that multiplications with zero coefficients are avoided and that saving in computer storage could be realized. Consider the following system of equations in matrix notation

$$\begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ K_{21} & K_{22} & \cdots & K_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ K_{n1} & K_{n2} & \cdots & K_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{4.17}$$

The Gaussian procedure solves one of the set of n equations (in this case x_1) in terms of all the other remaining unknowns $x_2 \cdots x_n$, then substituting this pivotal equation into the remaining $n-1$ equations. Thus x_1 has been eliminated from the last $n-1$ equations by $K_{lj} = K_{lj} - (\frac{K_{l1}}{K_{11}})K_{1j}$, $j = 1,n$ for $l = 2,n$ equations. Then it is repeated and x_2 is eliminated from each of the remaining $n-2$ equations. This procedure is duplicated for the all unresolved equations and leads to reduced Gauss equations as

$$\begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ 0 & K_{22} & \cdots & K_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & K_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{4.18}$$

The set of operations which reduce the original coefficients matrix $\boldsymbol{K}$ to the above triangular matrix form is referred to as *forward reduction*. Then the x_n value can be obtained from the n^{th} equation, the remaining components of the $\boldsymbol{x}$ vector can be solved by working backward. A complete solution process of the Gaussian elimination can be summarized as

Forward reduction

For $i = 1,n$-1 loop through rows $l = i+1,n$

$$K_{lj} = K_{lj} - \left(\frac{K_{li}}{K_{ii}}\right) K_{ij} \qquad j = i, n \tag{4.19a}$$

$$b_l = b_l - \left(\frac{K_{li}}{K_{ii}}\right) b_i \tag{4.19b}$$

Back Substitution

For $i = n, 1$

$$b_i = \frac{b_i - \left[\sum_{j=i+1}^{n} K_{ij} x_j\right]_{i<n}}{K_{ii}} \qquad x_j = b_j \tag{4.20}$$

For a symmetric matrix $\boldsymbol{K}$, the solution for *forward reduction* can be further modified and only the upper triangle coefficients of the $\boldsymbol{K}$ matrix are stored and operated on during forward reduction.

Forward reduction

For $i = 1, n-1$ loop through rows $l = i+1, n$

$$K_{lj} = K_{lj} - \left(\frac{K_{il}}{K_{ii}}\right) K_{ij} \qquad j = l, n \tag{4.21a}$$

$$b_l = b_l - \left(\frac{K_{il}}{K_{ii}}\right) b_i \tag{4.21b}$$

4.3.2 Standard Cholesky factorization

The standard Cholesky factorisation is only used, when the coefficient matrix $\boldsymbol{K}$ is a symmetric, positive definite matrix. In the standard Cholesky factorisation the coefficient matrix is decomposed as

$$\boldsymbol{K} = \boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{U}^T\boldsymbol{U} \tag{4.22a}$$

or

$$\boldsymbol{K}=\begin{bmatrix} L_{11} & 0 & \cdots & 0 \\ L_{21} & L_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ L_{n1} & L_{n2} & \cdots & L_{nn} \end{bmatrix}\begin{bmatrix} L_{11} & L_{2n} & \cdots & L_{n1} \\ 0 & L_{22} & \cdots & L_{n2} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & L_{nn} \end{bmatrix} \qquad 4.22b$$

where $\boldsymbol{L}$ is a lower triangular matrix and $\boldsymbol{U}$ is an upper triangular matrix. When the matrix $\boldsymbol{K}$ can be expressed in the factorized form of (4.22), the solution process can be derived as

$$\boldsymbol{K}\boldsymbol{x}=\boldsymbol{L}\left(\boldsymbol{L}^{T}\boldsymbol{x}\right)=\boldsymbol{L}\bar{\boldsymbol{b}}=\boldsymbol{b} \qquad 4.23a$$

$$\bar{\boldsymbol{b}}=\boldsymbol{L}^{-1}\boldsymbol{b}=\left(\boldsymbol{U}^{T}\right)^{-1}\boldsymbol{b} \qquad 4.23b$$

$$\boldsymbol{L}^{T}\boldsymbol{x}=\bar{\boldsymbol{b}} \qquad 4.23c$$

and

$$\boldsymbol{x}=\left(\boldsymbol{L}^{T}\right)^{-1}\bar{\boldsymbol{b}} \qquad 4.23d$$

For a general $n\times n$ matrix $\boldsymbol{K}$, the Cholesky factorization (4.22a) process can be carried out with column-by-column method as

Step 1 $$K_{11}=\left(K_{11}\right)^{\frac{1}{2}} \qquad 4.24a$$

For columns $j=2,n$ set

Step 2 $$K_{ij}=\frac{K_{ij}-\left[\sum_{l=1}^{i-1}K_{li}K_{lj}\right]_{i>1}}{K_{ii}} \qquad i=1,j-1 \qquad 4.24b$$

Step 3 $$K_{jj}=\left(K_{jj}-\sum_{l=1}^{j-1}K_{lj}^{2}\right)^{\frac{1}{2}} \qquad i=j \qquad 4.24c$$

For equation (4.23b), using *forward substitution*, the process can be set as

For $j=1,n$

$$b_{j}=\frac{b_{j}-\left[\sum_{l=1}^{j-1}K_{li}b_{l}\right]_{j>1}}{K_{jj}} \qquad 4.25$$

The equivalent algorithm for *backward substitution* of (4.23d) could be written as

For columns $j = n,1$

$$b_j = \frac{b_j}{K_{jj}} \tag{4.26a}$$

$$b_l = b_l - K_{lj} b_j \qquad l = j-1,1 \text{ and } j > 1 \tag{4.26b}$$

For the standard Cholesky factorisation, it can be shown that the coefficients of K must be positive definite, otherwise it is impossible to take the square roots in equation (4.24a) and (4.24c).

4.3.3 Modified Cholesky factorization

For some nonlinear solution procedures, the coefficient matrix $\boldsymbol{K}$ may be not positive definite. Therefore, the modified Cholesky factorisation method is considered as a preferred option. The coefficient matrix $\boldsymbol{K}$ can be decomposed as

$$\boldsymbol{K} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T \tag{4.27a}$$

or

$$\boldsymbol{K} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ L_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ L_{n1} & \cdots & L_{n(n-1)} & 1 \end{bmatrix} \begin{bmatrix} D_{11} & 0 & \cdots & 0 \\ 0 & D_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & D_{nn} \end{bmatrix} \begin{bmatrix} 1 & L_{12} & \cdots & L_{1n} \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & L_{(n-1)n} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \tag{4.27b}$$

where $\boldsymbol{D}$ is a non-singular diagonal matrix and $\boldsymbol{L}$ is an unit lower triangular matrix. For a non-symmetric matrix $\boldsymbol{K}$, $\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$ factorisation is replaced by $\boldsymbol{L}\boldsymbol{D}\boldsymbol{U}$, where $\boldsymbol{U}$ is an unit upper triangular matrix. The solution procedure (4.16) can be expressed as

$$\boldsymbol{K}\boldsymbol{x} = \boldsymbol{L}\left(\boldsymbol{D}\boldsymbol{L}^T\boldsymbol{x}\right) = \boldsymbol{L}\bar{\boldsymbol{b}} = \boldsymbol{b} \tag{4.28a}$$

where

$$\bar{\boldsymbol{b}} = \boldsymbol{L}^{-1}\boldsymbol{b} \tag{4.28b}$$

Equation (4.28b) involves forward substitution and it is the same as for equation (4.23b). Equation (4.28a) can be further decomposed as

$$\boldsymbol{D}\boldsymbol{L}^T\boldsymbol{x} = \boldsymbol{D}\bar{\bar{\boldsymbol{b}}} = \bar{\boldsymbol{b}} \qquad 4.28c$$

So that

$$\bar{\bar{\boldsymbol{b}}} = \boldsymbol{D}^{-1}\bar{\boldsymbol{b}} \qquad 4.28d$$

Following back substitution the solution vector $\boldsymbol{x}$ can be obtained as

$$\boldsymbol{x} = (\boldsymbol{L}^T)^{-1}\bar{\bar{\boldsymbol{b}}} \qquad 4.28e$$

The complete procedure of $\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T$ factorisation in equation (4.28) can be implemented using column-by-column methods as

For columns j=2,n

$$K_{ij} = K_{ij} - \left[\sum_{l=1}^{i-1} K_{lj}K_{li}\right]_{i>1} \qquad i = 1, j-1 \qquad 4.29a$$

$$K_{ij} = \frac{K_{ij}}{K_{ii}} \qquad 4.29b$$

$$K_{jj} = K_{jj} - \sum_{l=1}^{j-1} K_{lj}^2 K_{ll} \qquad i = j \qquad 4.29c$$

Forward substitution of equation (4.28b) can be written explicitly as,

For columns j=2,n

$$b_j = b_j - \left[\sum_{l=1}^{j-1} K_{lj}b_l\right]_{j>1} \qquad 4.29d$$

Back substitution of equation (4.28d) and (4.28e) can be expressed as,

For $i = 1, n$ set

$$b_i = \frac{b_i}{K_{ii}} \qquad 4.29e$$

For each columns j=n,2

$$b_l = b_l - K_{lj}b_j \qquad l = j-1, 1 \qquad 4.29f$$

By using the modified Cholesky factorization, the amount of data required for storage is dramatically reduced, for example, for a symmetric stiffness matrix $\boldsymbol{K}$ only one half of the $\boldsymbol{K}$ matrix needs to be stored, thus halving the arithmetic operations and memory requirements. As demonstrated in Chapter 2, the stiffness matrix $\boldsymbol{K}$ for the discretized finite element equations of Lagrangian flow is well banded, therefore we can reduce the required storage by using a skyline storage scheme, which avoids unnecessary operation on zero elements above the skyline. A detailed pictorial representation of the main step for the modified Cholesky factorization is given by Crisfield [4.14], and is shown in Figure 4.3.

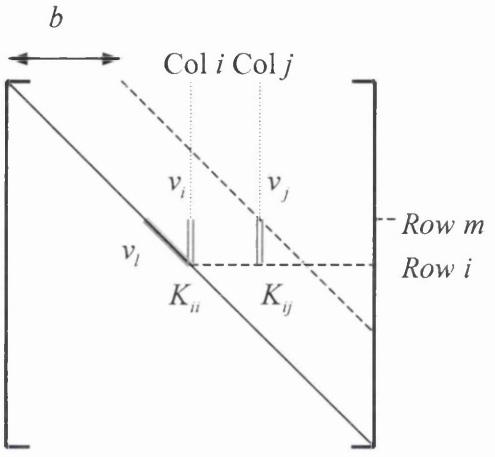


Figure 4.3(a)
Banded modified Cholesky factorization

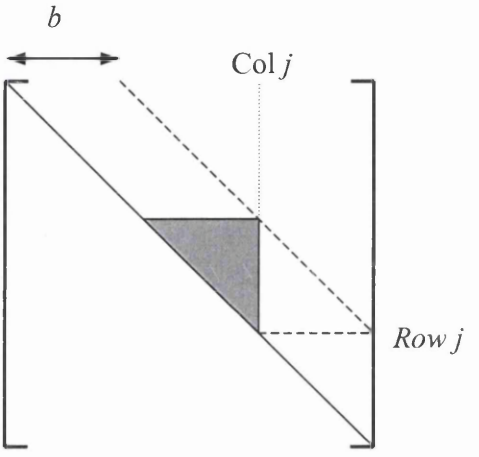


Figure 4.3(b)
Coefficient area for reduction of column j

In Figure 4.3(a)(b), the coefficients of the upper triangle of $\boldsymbol{K}$ are in a perfectly banded form between the diagonal line and dash line, where term b stands for the half bandwidth of matrix $\boldsymbol{K}$. The term $\boldsymbol{v}_i$ is a truncated column vector through element K_{ii} starting at row m and ending at row $i-1$ (included) and $\boldsymbol{v}_j$ is an equivalent vector through K_{ij}. The starting row m for a perfectly banded matrix $\boldsymbol{K}$ can be defined by

$$m = \max\left(1, \quad j+1-b\right) \tag{4.30}$$

Now the forward factorization in equations (4.29a)-(4.29c) are modified by changing the start of the summations from $l=1$ to $l=m$ as

$$K_{ij} = K_{ij} - \left[\sum_{l=m}^{i-1} K_{lj} K_{li}\right]_{i>m} = K_{ij} - \boldsymbol{v}_i^T \boldsymbol{v}_j \qquad i = m, j-1 \qquad 4.31a$$

$$K_{ij} = \frac{K_{ij}}{K_{ii}} \qquad 4.31b$$

$$K_{jj} = K_{jj} - \sum_{l=m}^{j-1} K_{lj}^2 K_{ll} \qquad i = j \qquad 4.31c$$

where an inner product of $\boldsymbol{v}_i^T \boldsymbol{v}_j$ in equation (4.31a) is carried out on the upper part of the truncated column vectors, that avoids unnecessary zero product operations above the skyline. The complete reduction of a column j will be only limited at the shaded triangular area, as shown in Figure 4.3(b). However for some finite element meshes, the resulting coefficient of $\boldsymbol{K}$ will be less perfectly banded; even for a regular mesh the stiffness coefficients are not in a perfectly banded form. Therefore, the starting row m defined in equation (4.30) is usually replaced by

$$m = \max\left(m(j),\ \ m(i)\right) \qquad 4.32$$

where $m(i)$ and $m(j)$ are the starting row for vectors i and j respectively, as shown in Figure 4.4

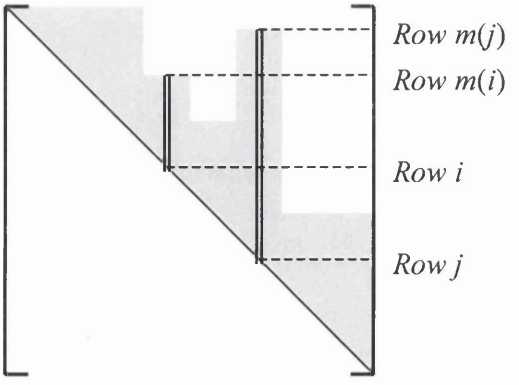


Figure 4.4 Active column heights

Since the procedure is calculated column by column, the upper triangular coefficients matrix $\boldsymbol{K}$ can be stored as a one–dimensionial array, with two different active column storage schemes as illustrated in Figure 4.5.

$$\boldsymbol{K}=\begin{bmatrix} 1 & 2 & 4 & 7 & & & & & & \\ & 3 & 5 & 8 & & & 17 & & & \\ & & 6 & 9 & 11 & & 18 & & & \\ & & & 10 & 12 & 14 & 19 & & & \\ & & & & 13 & 15 & 20 & & & \\ & & & & & 16 & 21 & 23 & 26 & 30 \\ & & & & & & 22 & 24 & 27 & 31 \\ & & & & & & & 25 & 28 & 32 \\ & & & & & & & & 29 & 33 \\ & & & & & & & & & 34 \end{bmatrix}$$

$$\boldsymbol{K}=\begin{bmatrix} 1 & 12 & 13 & 15 & & & & & & \\ & 2 & 14 & 16 & & & 22 & & & \\ & & 3 & 17 & 18 & & 23 & & & \\ & & & 4 & 19 & 20 & 24 & & & \\ & & & & 5 & 21 & & & & \\ & & & & & 6 & 25 & 26 & 28 & 31 \\ & & & & & & 7 & 27 & 29 & 32 \\ & & & & & & & 8 & 30 & 33 \\ & & & & & & & & 9 & 34 \\ & & & & & & & & & 10 \end{bmatrix}$$

diag=[1, 3, 6, 10, 13 16 , 22 ,25, 29 ,34]

index=[12, 12, 13, 15, 18, 20, 22, 26, 28, 31, 35, 1, 1, 2, 1, 2, 3, 3, 4, 4, 5, 2, 3, 4, 6, 6, 7, 6, 7, 8, 6, 7, 8, 9]

(a) Active column storage scheme type 1 (b) Active column storage scheme type 2

Figure 4.5 Example $\boldsymbol{K}$ matrix with one-dimensional storage.

The active column storage scheme type 1 as shown in Figure 4.5(a) is used for a direct profile solver. The numbering of the stiffness coefficients relates to all the elements below the 'skyline' including those with stiffness coefficients that are zero. For example, in column 7 the coefficient in position 20 is zero, but it must be included. In general, the space will be filled during the factorisation. The location of the diagonal element is recorded in an integer vector ***diag***(n), where n is the dimension of solution. In addition, an integer array $\boldsymbol{Ih}(n)$ is opened to record each column height, i.e. the number of elements under the 'sky-line' in each active column, which includes the diagonal element.

The storage scheme type 2 in Figure 4.5(b) is more suitable for an iterative solver. The numbering of the stiffness coefficients relates to all the element below the 'sky-line', but excluding those with zero coefficients. An integer array ***index***(*nstif*), where the term *nstif* is the size of the one-dimensional stiffness array, is created in Figure

4.5(b), the first 10 elements of ***index*** record the position of the first non-zero coefficient of each active column in the stiffness array, the eleventh element is assigned with number (*nstif*+1), the remaining terms record the location of corresponding unknown variables in the active global solution, which excludes the diagonal position of the corresponding unknown variables.

Both schemes shown in Figure 4.5(a)(b) are only set for a symmetric system. With an un-symmetric one, the lower triangular part will be also recorded in the same manner, but it adopts an active row storage scheme.

4.4 Iterative solution of linear systems

Originally, the usage of iterative methods was restricted to systems related to elliptic partial differential equations, discretized by finite difference techniques. For the problems related to various finite element modelling, practitioners preferred the usage of direction solution techniques, mainly efficient variants of the Gaussian elimination, because of the lack of robustness of iterative methods for large classes of matrices. Until the end of 1980s, almost none of the big commercial finite element analysis packages included iterative solution techniques. The Krylov subspace methods, which appeared in the early 1950s and rapidly developed during the recent two decades, have changed the landscape of iterative methods dramatically. Some of the Krylov subspace methods have been widely accepted as powerful tools for the iterative solution of very large sparse linear systems. In this section, we will highlight a number of iterative methods built on the Krylov subspace in detail.

4.4.1 Krylov subspace methods

Krylov subspace methods started in the early 1950s with the introduction of the conjugate gradients methods by Hestnes and Stiefel [4.15] and the Lanczos algorithm for linear equations [4.16]. These methods are designed to construct an approximate solution in the so-called Krylov subspace. For the linear system given by equation

(4.16), with a large, sparse, non-singular n by n stiffness matrix $\boldsymbol{K}$, then the standard Richardson iteration produces the i^{th} iteration solution as

$$\boldsymbol{x}_i = (\boldsymbol{I} - \boldsymbol{K})\boldsymbol{x}_{i-1} + \boldsymbol{b} \tag{4.33}$$

Which generates the approximate solution in the shifted Krylov subspace

$$\boldsymbol{x}_i \approx \boldsymbol{x}_0 + \boldsymbol{\kappa}_i(\boldsymbol{K}, \boldsymbol{r}_0) = \boldsymbol{x}_0 + span\left\{\boldsymbol{r}_0, \boldsymbol{K}\boldsymbol{r}_0, \boldsymbol{K}^2\boldsymbol{r}_0 ..., \boldsymbol{K}^{i-1}\boldsymbol{r}_0\right\} \tag{4.34}$$

where $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{K}\boldsymbol{x}_0$, with an arbitiary initial guess vector $\boldsymbol{x}_0$. With relativly little additional work, more efficient approximation solutions can be constructed from the Krylov subspace, which leads to Krylov subspace projection methods. This type of iteration methods includes the Conjugate Gradient (CG), Generalized Minimal Residual (GMRES) BiConjugate Gradient (BiCG), BiConjugate Gradient Stabilized (Bi-CGSTAB).

4.4.2 Conjugate Gradient Method (CG)

The Conjugate Gradient method was presented by Hestenes and Stiefel [4.15] in 1952. It was initially used as a direct solver and later considered as a truly iterative method for solving sparse, symmetric, positive definite linear systems [4.17]-[4.20], which are too large to be handled by direct methods such as the Cholesky decomposition. For a symmetric, positive definite $\boldsymbol{K}$, the CG method minimizes the so-called $\boldsymbol{K}$-norm, $\|\boldsymbol{x}_i - \boldsymbol{x}\|_{\boldsymbol{K}}^2 = (\boldsymbol{x}_i - \boldsymbol{x}, \boldsymbol{K}(\boldsymbol{x}_i - \boldsymbol{x}))$, for $\boldsymbol{x}_i$ that are in the Krylov subspace $\boldsymbol{\kappa}_i(\boldsymbol{K}, \boldsymbol{r}_0)$. For some PDE problems this norm is also called an energy norm, which has a physical meaning. Another feature of the CG is that the residual calculated in the current iteration is orthogonal to the space of previously generated residuals. It is therefore mathematically equivalent to the Full Orthogonalization Method (FOM) [4.21]. The essence of the CG method is outlined as follow.

To solve equation (4.16), we start with the initial estimate $\boldsymbol{x}_0$, then the residual of the first approximate solution $\boldsymbol{r}_0$ can be set by

$$\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{K}\boldsymbol{x}_0 \tag{4.35}$$

The first direction of minimization, $\boldsymbol{P}_0 = \boldsymbol{r}_0$, is a steepest descent direction of the quadratic function $\phi(\boldsymbol{x})$, which is defined as

$$\phi(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{K}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{b} \tag{4.36}$$

A sequence of approximate solutions $\boldsymbol{x}_{i+1}$ can be obtained from $\boldsymbol{x}_i$ along a search direction $\boldsymbol{P}_i$ with a step length as

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \boldsymbol{P}_i \tag{4.37}$$

where the step length parameter α_i is determined by minimization of the quadratic function $\phi(\boldsymbol{x}, \alpha_i)$ with respect to α_i

$$\frac{\partial \phi(\boldsymbol{x}_i + \alpha_i \boldsymbol{P}_i)}{\partial \alpha_i} = 0 \tag{4.38}$$

and

$$\alpha_i = \frac{(\boldsymbol{P}_i, \boldsymbol{r}_i)}{(\boldsymbol{P}_i, \boldsymbol{K}\boldsymbol{P}_i)} \tag{4.39}$$

where $(\ ,\ )$ denotes a vector inner product. The term $\boldsymbol{r}_i = \boldsymbol{b} - \boldsymbol{K}\boldsymbol{x}_i$ is the residual of the i^{th} approximate solution $\boldsymbol{x}_i$. The residual of the $(i+1)^{th}$ approximate solution $\boldsymbol{r}_{i+1}$ can be calculated by

$$\boldsymbol{r}_{i+1} = \boldsymbol{b} - \boldsymbol{K}\boldsymbol{x}_{i+1} = \boldsymbol{r}_i - \alpha_i \boldsymbol{K}\boldsymbol{P}_i \tag{4.40}$$

Now a new search direction $\boldsymbol{P}_{i+1}$ at the $(i+1)^{th}$ iteration is constructed by

$$\boldsymbol{P}_{i+1} = \boldsymbol{r}_{i+1} + \beta_i \boldsymbol{P}_i \tag{4.41}$$

Thus the consequences of above relation is that

$$(\boldsymbol{P}_i, \boldsymbol{r}_i) = ((\boldsymbol{r}_i + \beta_{i-1}\boldsymbol{P}_{i-1}), \boldsymbol{r}_i) = (\boldsymbol{r}_i, \boldsymbol{r}_i) \tag{4.42}$$

where it uses the property $\left(\boldsymbol{P}_i, \boldsymbol{r}_j\right) = 0$ for $i < j$, then substituting (4.42) into equation (4.39) α_i becomes

$$\alpha_i = \frac{\left(\boldsymbol{r}_i, \boldsymbol{r}_i\right)}{\left(\boldsymbol{P}_i, \boldsymbol{K}\boldsymbol{P}_i\right)} \tag{4.43}$$

The parameter β_i in equation (4.41) is determined by imposing the conjugate condition $\left(\boldsymbol{P}_{i+1}, \boldsymbol{K}\boldsymbol{P}_i\right) = 0$ and the property $\left(\boldsymbol{r}_i, \boldsymbol{r}_j\right) = 0$ for $i \neq j$.

$$\beta_i = -\frac{\left(\boldsymbol{r}_{i+1}, \boldsymbol{K}\boldsymbol{P}_i\right)}{\left(\boldsymbol{P}_i, \boldsymbol{K}\boldsymbol{P}_i\right)} = \frac{\left(\boldsymbol{r}_{i+1}, \boldsymbol{r}_{i+1}\right)}{\left(\boldsymbol{r}_i, \boldsymbol{r}_i\right)} \tag{4.44}$$

The details of the above conjugate gradient algorithm can be summarized in Figure 4.6

Compute $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{K}\boldsymbol{x}_0$ *from initial guess* $\boldsymbol{x}_0$ *and set* $\boldsymbol{P}_0 = \boldsymbol{r}_0$
Do $i = 0, 1, 2, \ldots.$ *miter*

(a) Update solution

$$\boldsymbol{w}_i = \boldsymbol{K}\boldsymbol{P}_i$$

$$\alpha_i = \frac{\left(\boldsymbol{r}_i, \boldsymbol{r}_i\right)}{\left(\boldsymbol{P}_i, \boldsymbol{w}_i\right)}$$

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \Delta\boldsymbol{x}_i = \boldsymbol{x}_i + \alpha_i \boldsymbol{P}_i$$

$$\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \alpha_i \boldsymbol{w}_i$$

(b) Check convergence
If $\left\|\boldsymbol{r}_{i+1}\right\|_2 < \varepsilon_1 \left\|\boldsymbol{b}\right\|_2$ *or* $\left\|\Delta\boldsymbol{x}_i\right\|_2 < \varepsilon_2 \left\|\boldsymbol{x}_{i+1}\right\|_2$ *then stop;*
else continue

(c) Set new search direction

$$\beta_i = \frac{\left(\boldsymbol{r}_{i+1}, \boldsymbol{r}_{i+1}\right)}{\left(\boldsymbol{r}_i, \boldsymbol{r}_i\right)}$$

$$\boldsymbol{P}_{i+1} = \boldsymbol{r}_{i+1} + \beta_i \boldsymbol{P}_i$$

EndDo;

Figure 4.6 Conjugate Gradient algorithm

From Figure 4.6 it is seen that in addition to the matrix $\boldsymbol{K}$, four vectors $(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{w}, \boldsymbol{r})$ must be opened and stored in database for the CG algorithm, where ε_1 and ε_2 are the residual force and incremental displacement tolerances, respectively, normally set as ε_1=1.0e-5 and ε_2=1.0e-8.

For a symmetric, positive definite matrix $\boldsymbol{K}$, the error at the i^{th} iteration, in the CG algorithm, is limited by the well-known upper bound [4.22],

$$\|\boldsymbol{x}_i - \boldsymbol{x}\|_2 \leq 2\left(\frac{\sqrt{k}-1}{\sqrt{k}+1}\right)^i \|\boldsymbol{x}_0 - \boldsymbol{x}\|_2 \qquad 4.45$$

where $k = \lambda_{\max}(\boldsymbol{K}) / \lambda_{\min}(\boldsymbol{K})$ is the condition number of $\boldsymbol{K}$, and $\boldsymbol{x}_i$ denotes the i^{th} approximate solution. The upper bound defines the convergence behaviour for matrices $\boldsymbol{K}$ of which the eigenvalues are distributed rather homogeneously.

4.4.3 Preconditioned Conjugate Gradient Method (PCG)

The convergence rate of the Conjugate Gradient method depends on the eigenvalue distribution of the stiffness matrix $\boldsymbol{K}$, as described in equation (4.45). The convergence is faster, when the condition number of $\boldsymbol{K}$ is smaller or $\boldsymbol{K}$ has clustered eigenvalues. For ill-conditioned problems the convergence of the CG method might be slow, mainly due to round-off errors. For such problems, the conjugate directions are no longer exactly conjugate after some iteration. It is possible to accelerate the rate of convergence by the transformation of equation (4.16) such that the eigenvalue distribution of $\boldsymbol{K}$ is improved. This process is called preconditioning and changes solution $\boldsymbol{x}$ into $\tilde{\boldsymbol{x}}$ by a full rank pre-conditioner matrix $\boldsymbol{C}$.

$$\boldsymbol{x} = \boldsymbol{C}\tilde{\boldsymbol{x}} \qquad 4.46$$

Substituting equation (4.46) into (4.16) and pre-multiplying equation (4.16) by $\boldsymbol{C}^T$, gives then a transformed equation system,

$$\tilde{\boldsymbol{K}}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}} \qquad 4.47$$

where

$$\tilde{\boldsymbol{K}} = \boldsymbol{C}^T \boldsymbol{K} \boldsymbol{C}$$

$$\tilde{\boldsymbol{b}} = \boldsymbol{C}^T \boldsymbol{b}$$

The convergence rate of the CG method applied to equation (4.47) is now dependent on the eigenvalue distribution of the preconditioned matrix $\tilde{\boldsymbol{K}}$ rather than those of $\boldsymbol{K}$. The objective of the pre-conditioning is to choose a proper pre-conditioner matrix $\boldsymbol{C}$ such that the condition number of $\tilde{\boldsymbol{K}}$ is much smaller than that of the original matrix $\boldsymbol{K}$. Alternatively, $\boldsymbol{C}$ can be chosen such that the eigenvalues of $\tilde{\boldsymbol{K}}$ are clustered. Detailed discussion of preconditioning techniques will be set in section 4.4.6

The Preconditioned Conjugate Gradient algorithm can be simply derived following the previous steps. The CG method is applied to the preconditioned system (4.47) with the quadratic function defined by

$$\phi(\tilde{\boldsymbol{x}}) = \frac{1}{2} \tilde{\boldsymbol{x}}^T \tilde{\boldsymbol{K}} \tilde{\boldsymbol{x}} - \tilde{\boldsymbol{x}}^T \tilde{\boldsymbol{b}} \qquad 4.48$$

The i^{th} solution $\boldsymbol{x}_i$ and search direction $\boldsymbol{P}_i$ can be set by

$$\boldsymbol{x}_i = \boldsymbol{C} \tilde{\boldsymbol{x}}_i$$

$$\boldsymbol{P}_i = \boldsymbol{C} \tilde{\boldsymbol{P}}_i \qquad 4.49$$

Then the relation between the i^{th} residual $\tilde{\boldsymbol{r}}_i$ and $\boldsymbol{r}_i$ is given by

$$\tilde{\boldsymbol{r}}_i = \tilde{\boldsymbol{b}} - \tilde{\boldsymbol{K}} \tilde{\boldsymbol{x}}_i = \boldsymbol{C}^T (\boldsymbol{b} - \boldsymbol{K} \boldsymbol{C} \boldsymbol{C}^{-1} \boldsymbol{x}_i) = \boldsymbol{C}^T \boldsymbol{r}_i$$

$$\boldsymbol{r}_i = \boldsymbol{C}^{-T} \tilde{\boldsymbol{r}}_i \qquad 4.50$$

By minimization of quadratic function $\phi(\tilde{\boldsymbol{x}}, \alpha_i)$ with respect to α_i, we obtain,

$$\alpha_i = \frac{(\tilde{\boldsymbol{r}}_i, \tilde{\boldsymbol{r}}_i)}{(\tilde{\boldsymbol{P}}_i, \tilde{\boldsymbol{K}} \tilde{\boldsymbol{P}}_i)} = \frac{(\boldsymbol{r}_i, \boldsymbol{C} \boldsymbol{C}^T \boldsymbol{r}_i)}{(\boldsymbol{P}_i, \boldsymbol{K} \boldsymbol{P}_i)} = \frac{(\boldsymbol{r}_i, \boldsymbol{z}_i)}{(\boldsymbol{P}_i, \boldsymbol{K} \boldsymbol{P}_i)} \qquad 4.51$$

where we introduce an auxiliary vector $z_i = CC^T r_i$. For no preconditioning $CC^T = I$ and $z_i = r_i$. The parameter β_i and the new search direction P_{i+1} can be derived as,

$$\beta_i = \frac{(r_{i+1}, z_{i+1})}{(r_i, z_i)} \qquad 4.52$$

$$P_{i+1} = z_{i+1} + \beta_i P_i \qquad 4.53$$

The preconditioned conjugate gradient algorithm is summarized in Figure 4.7

Compute $r_0 = b - Kx_0$ *from initial guess* x_0,
set $z_0 = CC^T r_0$, $P_0 = z_0$
Do $i = 0,1,2,\ldots.$ *miter*

(a) Update solution

$w_i = KP_i$

$$\alpha_i = \frac{(r_i, z_i)}{(P_i, w_i)}$$

$x_{i+1} = x_i + \Delta x_i = x_i + \alpha_i P_i$

$r_{i+1} = r_i - \alpha_i w_i$

(b) Check convergence
If $\|r_{i+1}\|_2 < \varepsilon_1 \|b\|_2$ *or* $\|\Delta x_i\|_2 < \varepsilon_2 \|x_{i+1}\|_2$ *then stop;*
else continue

(c) Set new search direction

$z_{i+1} = CC^T r_{i+1}$

$$\beta_i = \frac{(r_{i+1}, z_{i+1})}{(r_i, z_i)}$$

$P_{i+1} = z_{i+1} + \beta_i P_i$

EndDo;

Figure 4.7 Pre-conditioned Conjugate Gradient algorithm

Comparing Figure 4.7 and Figure 4.6 it is seen that five vectors (x, P, w, r, z) and pre-conditioner matrix CC^T must be opened and stored in database for the PCG algorithm, with the exception of the matrix K. Here, we compare the conjugate

gradients preconditioned with incomplete Cholesky factorisation to the standard CG methods. The number of iterations for solving a symmetric linear system with solution dimension $n = 1137$ is shown in Table 4.1, where a residual tolerance $\varepsilon_1 = 1.0e^{-5}$ and displacement tolerance $\varepsilon_2 = 1.0e^{-8}$ is employed. It shows that with preconditioning the convergence rate is dramatically improved.

Method	*Number of iteration*
CG	1293
ICCG	284

Table 4.1 CG/ICCG comparison

4.4.4 Generalized Minimal Residual method (GMRES)

The Generalized Minimal Residual Method (GMRES) was proposed by Saad and Schultz[4.23] in 1986, and soon came to be preferred because of better numerical behaviour and lower cost, in terms of memory and arithmetic. The reason to develop GMRES was in order to solve the systems where the coefficient matrix $\boldsymbol{K}$ may not be positive real. In general GMRES is utilised to solve large un-symmetric semi-positive definite systems. The term semi-positive definite denotes that the majority of the eigenvalues are known to have real positive parts with some of the eigenvalues real part being equal to zero. The algorithm guarantees convergence when the eigenvalues of the coefficient matrix are all positive, therefore it is rationalized that GMRES can converge when most of the eigenvalues are positive.

GMRES is a projection method, which projects the solution $\boldsymbol{x}$ onto the m^{th} Krylov subspace κ_m with $\boldsymbol{v}_1 = \boldsymbol{r}_0 / \|\boldsymbol{r}_0\|_2$, where $\boldsymbol{r}_0$ is the initial residual vector. The base of the Krylov subspace κ_m is generated by the Arnoldi process [4.24], which is an orthogonormalizing algorithm using the Modified Gram-Schmidt procedure, (see Appendix 4.7.2). The dimension m of Krylov subspace κ_m is far less than the dimension of solution space $\boldsymbol{R}^n$, i.e. $m \ll n$. After the j^{th} iteration when the residual

is less than a tolerance value, the solution $\boldsymbol{x}_j$ is approximated by a linear combination of the base vectors of the m^{th} Krylov subspace as

$$\boldsymbol{x}_j = \boldsymbol{x}_0 + \boldsymbol{V}_j \boldsymbol{y}_j \qquad 4.54$$

where $\boldsymbol{V}_j = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots \boldsymbol{v}_j\}$ represents a set of orthogonal base vectors and $1 \le j \le m$, $m \ll n$, where n is the dimension of the solution. $\boldsymbol{y}_j^T = \{y_1, y_2, \ldots y_j\}$ is a vector with j unknown components to be determined. By minimizing the Euclidean residual norm $\|\boldsymbol{b} - \boldsymbol{K}\boldsymbol{x}_j\|_2$ over the Krylov subspace, the components of the vector $\boldsymbol{y}_j$ can be identified. An advantage of the GMRES is its guarantee to compute the approximate solution with a minimum residual norm, but the price to be paid is that additional base vectors must be stored for each iteration, which means that more iterations are performed, more base vectors have to be stored, and also the overhead cost per iteration increases linearly. A partial solution to the problem is to restart the solution procedure after m iterations, to keep the memory requirements and the work per iteration limited. The difficulty is to choose an appropriate value for m, too small a value may cause slow convergence, a larger value may require unnecessary storage space. The popular choices of the m value tend to be in the range of 5 to 30 [4.26].

The basic GMRES algorithm is described in detail in Reference [4.25]. Figure 4.8 shows the pseudocode of the algorithm.

(1) *Compute initial residual* $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{K}\boldsymbol{x}_0$ *for some initial guess* $\boldsymbol{x}_0$

$$\beta = \|\boldsymbol{r}_0\|_2; \qquad \boldsymbol{v}_1 = \frac{\boldsymbol{r}_0}{\beta}; \qquad \hat{b}_1 = \beta$$

(2) *Define the* $(m+1)\times m$ *matrix* $\bar{\boldsymbol{H}}_m = \{h_{ij}\}_{1\le i\le m+1, 1\le j\le m}$. *Set* $\bar{\boldsymbol{H}}_m = 0$

Construct orthogonal base vectors

(3) **Do** $j = 1, \ldots, m$

(4) Solve $\boldsymbol{w}_j = \boldsymbol{K}\boldsymbol{v}_j$

(5) **Do** $i = 1, \ldots, j$

(6) $h_{i,j} = (\boldsymbol{w}_j, \boldsymbol{v}_i)$

(7) $\boldsymbol{w}_j = \boldsymbol{w}_j - h_{i,j}\boldsymbol{v}_i$

(8) **EndDo**

(9) $h_{j+1,j} = \|w_j\|_2$ *if* $h_{j+1,j} = 0$, *set m=j goto 21*

(10) $v_{j+1} = \dfrac{w_j}{h_{j+1,j}}$

(11) **Do** $i = 1,..., j-1$

(12) $\alpha = h_{i,j}$

(13) $h_{i,j} = c_i \alpha + s_i h_{i+1,j}$

(14) $h_{i+1,j} = -s_i \alpha + c_i h_{i+1,j}$

(15) **EndDo**

(16) $\delta = \sqrt{h_{j,j}^2 + h_{j+1,j}^2}$; $c_j = \dfrac{h_{j,j}}{\delta}$; $s_j = \dfrac{h_{j+1,j}}{\delta}$

(17) $h_{j,j} = \delta$; $h_{j+1,j} = 0$

(18) $\hat{b}_{j+1} = -s_j r_j$; $\hat{b}_j = c_j r_j$

(19) *Check if* $\rho = \left|\hat{b}_{j+1}\right| < \varepsilon_1 \|b\|_2$ *then* $m = j$ *goto* (21)

(20) **EndDo**

(21) Compute y_m the minimizer of $\left\|\hat{b}_m - \bar{R}_m y_m\right\|_2$ and

$x_m = x_0 + \Delta x_m = x_0 + V_m y_m$

(22) *Check displacement norm*

If $\|\Delta x_m\|_2 < \varepsilon_2 \|x_m\|_2$ *then stop; else* $x_0 \Leftarrow x_m$ *goto step* (1)

Figure 4.8 GMRES algorithm

The solution steps in Figure 4.8 can be explained as follows;

Step (1) With initial guess value x_0, the initial residual r_0 is computed, where β is the norm of the residual vector and v_1 is the normalized residual vector, from which we start to build the Krylov subspace.

Step (2) Initialise the Hessenberg matrix $\bar{H}_m$ with dimension $(m+1)\times m$. The matrix $\bar{H}_m$ after Arnoldi process has a direct relation with coefficient matrix K. By removing the last row of $\bar{H}_m$, H_m preserves the major eigenvalues of K.

Step (3-20) The procedure in steps (3)-(10) is the Arnoldi process, which uses the modified Gram- Schmidt orthogonormalizing method (see Appendix 4.7.1) applied to the Krylov subspace. The vectors $v_1, v_2, \ldots v_m$ generated in Arnoldi's process form the orthonormal basis of the m^{th} Krylov subspace. At the same time the Hessenberg

matrix $\bar{\boldsymbol{H}}_m$ is defined. Set $\boldsymbol{V}_m = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots \boldsymbol{v}_m\}$ with dimension $n \times m$ and $\boldsymbol{H}_m$ as $\bar{\boldsymbol{H}}_m$ minus the last row, then the relation between the coefficient matrix $\boldsymbol{K}$ and Hessenberg matrix $\boldsymbol{H}_m$ is given by

$$\boldsymbol{K}\boldsymbol{V}_m = \boldsymbol{V}_m\boldsymbol{H}_m + \boldsymbol{w}_m\boldsymbol{e}_m^T = \boldsymbol{V}_{m+1}\bar{\boldsymbol{H}}_m \qquad 4.55$$

$$\boldsymbol{V}_m^T\boldsymbol{K}\boldsymbol{V}_m = \boldsymbol{H}_m \qquad 4.56$$

This is why $\boldsymbol{H}_m$ preserves the major eigenvalues of the coefficient matrix. A breakdown can occur in the Arnoldi loop when $h_{j+1,j} = 0$ at step (9), in this situation, the next Arnoldi vector cannot be generated. However in this situation, the residual vector is zero, i.e. the algorithm will deliver the exact solution at this step. Such a breakdown is called "lucky breakdown", but it rarely occurs in real situations due to machine round off errors.

The procedure in step (11)-(19) is the QR decomposition algorithm, which transforms the Hessenberg matrix into an upper triangular form by using plane rotations. (See Appendix 4.7.3 for more details.)

Step (19) Check the residual of the j^{th} iteration, as we know from Appendix 4.7.3, the residual norm of the j^{th} iteration is defined by the $(j+1)^{th}$ component of the $\hat{\boldsymbol{b}}_j$ vector. If its norm is less than the force tolerance then set $m = j$ and go to step (21).

Step (21) After a set of orthogonal basis vectors $\boldsymbol{V}_m$ is obtained, the solutions of the system $\boldsymbol{x}$ is approximated by

$$\boldsymbol{x} \approx \boldsymbol{x}_m = \boldsymbol{x}_0 + \boldsymbol{V}_m\boldsymbol{y}_m \qquad 4.57$$

where $\boldsymbol{y}_m$ is an m-dimension unknown vector, which can be defined through minimizing the residual norm function $f(\boldsymbol{y})$

$$f(\boldsymbol{y}_m) = \|\boldsymbol{b} - \boldsymbol{K}\boldsymbol{x}_m\|_2 = \|\boldsymbol{b} - \boldsymbol{K}(\boldsymbol{x}_0 + \boldsymbol{V}_m\boldsymbol{y}_m)\|_2 \qquad 4.58$$

Using the relation stated in equation (4.55)

$$\begin{aligned} \boldsymbol{b}-\boldsymbol{K}\boldsymbol{x}_m &= \boldsymbol{b}-\boldsymbol{K}\left(\boldsymbol{x}_0+\boldsymbol{V}_m\boldsymbol{y}_m\right) \\ &= \boldsymbol{r}_0-\boldsymbol{K}\boldsymbol{V}_m\boldsymbol{y}_m \\ &= \beta\boldsymbol{v}_1-\boldsymbol{V}_{m+1}\bar{\boldsymbol{H}}_m\boldsymbol{y}_m \\ &= \boldsymbol{V}_{m+1}\left(\beta\boldsymbol{e}_1-\bar{\boldsymbol{H}}_m\boldsymbol{y}_m\right) \end{aligned} \qquad 4.59$$

Since the column vectors of $\boldsymbol{V}_{m+1}$ are orthogonal then

$$\begin{aligned} f\left(\boldsymbol{y}_m\right) = \left\|\boldsymbol{b}-\boldsymbol{K}\boldsymbol{x}_m\right\|_2 &= \left\|\beta\boldsymbol{e}_1-\bar{\boldsymbol{H}}_m\boldsymbol{y}_m\right\|_2 \\ &= \left\|\hat{\boldsymbol{b}}_m-\bar{\boldsymbol{R}}_m\boldsymbol{y}_m\right\|_2 \end{aligned} \qquad 4.60$$

By minimizing of the function $f\left(\boldsymbol{y}_m\right)$ the following $\left(m+1\right)$ equations are obtained

$$\bar{\boldsymbol{R}}_m\boldsymbol{y}_m = \hat{\boldsymbol{b}}_m \qquad 4.61$$

Since $\bar{\boldsymbol{R}}_m$ is an upper triangular matrix, $\boldsymbol{y}_m$ can be solved readily by using back substitution.

Step (22) Check the convergence. If the incremental displacement norm is not less than the tolerance, set $\boldsymbol{x}_0 = \boldsymbol{x}_m$ and start back at the beginning, clearing all storage when restarting. A common problem with restarting is the possibility of stagnation when the matrix is not positive definite. Stagnation is where the residual does not tend to zero, virtually staying the same. One solution is to apply a pre-conditioner matrix to the system.

4.4.5 Bi-conjugate Gradient Stabilized method (Bi-CGSTAB)

Faber and Manteuffel [4.32] in 1984 discovered that it is in general not possible to construct an optimal solution in the Krylov subspace for an un-symmetric $\boldsymbol{K}$, since the residual vectors cannot be made orthogonal with short recurrences. One alternative to solve this problem is to generate two mutually orthogonal sequences of residual vectors, which replace the original orthogonal sequences of residuals as

$$\kappa_m\left(\boldsymbol{K}, v_1\right) = span\left\{v_1, \boldsymbol{K}v_1, ..., \boldsymbol{K}^{m-1}v_1\right\} \quad 4.62a$$

and

$$\zeta_m\left(\boldsymbol{K}^T, \bar{v}_1\right) = span\left\{\bar{v}_1, \boldsymbol{K}^T\bar{v}_1, ..., \left(\boldsymbol{K}^T\right)^{m-1}\bar{v}_1\right\} \quad 4.62b$$

Equation (4.62a) retains the original coefficient matrix $\boldsymbol{K}$, while equation (4.62b) uses the $\boldsymbol{K}$ transpose matrix. Both Bi-CG and Bi-CGSTAB methods are developed based on the biorthogonal bases of the two subspaces.

Bi-CG algorithm is a projection on $\kappa_m\left(\boldsymbol{K}, v_1\right)$ orthogonal to $\zeta_m\left(\boldsymbol{K}^T, \bar{v}_1\right)$, where as usual $v_1 = \boldsymbol{r}_0 / \left\|\boldsymbol{r}_0\right\|_2$. The vector $\bar{v}_1$ is arbitrary, provided $\left(v_1, \bar{v}_1\right) \neq 0$, and is often set as $\bar{v}_1 = v_1$. If there is a dual system $\boldsymbol{K}^T\bar{\boldsymbol{x}} = \bar{\boldsymbol{b}}$ to be solved with $\boldsymbol{K}^T$, then $\bar{v}_1$ is the unit residual vector for the dual system. This dual system is often ignored in the formulation of the algorithm.

In the Bi-CG method, the approximations are constructed in such a way that the residual $\boldsymbol{r}_i$ is orthogonal with respect to another row of vector $\bar{\boldsymbol{r}}_0, \bar{\boldsymbol{r}}_1, \ldots \bar{\boldsymbol{r}}_{i-1}$, and vice versa $\bar{\boldsymbol{r}}_i$ is orthogonal with respect to $\boldsymbol{r}_0, \boldsymbol{r}_1, \ldots \boldsymbol{r}_{i-1}$. The algorithm comprises two 3-terms recurrence relations in the same manner as for the derivation of the Conjugate Gradient method. In the Bi-CG algorithm the corresponding residuals are updated as

$$\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \alpha_i \boldsymbol{K}\boldsymbol{P}_i \quad 4.63a$$

$$\bar{\boldsymbol{r}}_{i+1} = \bar{\boldsymbol{r}}_i - \alpha_i \boldsymbol{K}^T\bar{\boldsymbol{P}}_i \quad 4.63b$$

where two new search directions are updated using the resulting residual vectors

$$\boldsymbol{P}_{i+1} = \boldsymbol{r}_{i+1} + \beta_i \boldsymbol{P}_i \quad 4.64a$$

$$\bar{\boldsymbol{P}}_{i+1} = \bar{\boldsymbol{r}}_{i+1} + \beta_i \bar{\boldsymbol{P}}_i \quad 4.64b$$

The coefficients α_i and β_i in equation (4.64) are defined as

$$\alpha_i = \frac{\left(\bar{\boldsymbol{r}}_i, \boldsymbol{r}_i\right)}{\left(\bar{\boldsymbol{P}}_i^T, \boldsymbol{K}\boldsymbol{P}_i\right)} \quad 4.65$$

$$\beta_i = \frac{\left(\bar{r}_{i+1}, r_{i+1}\right)}{\left(\bar{r}_i, r_i\right)} \qquad 4.66$$

The whole procedure of the preconditioned Bi-CG algorithm is described in Figure 4.9 .

Compute $r_0 = b - Kx_0$ *from initial guess* x_0, $\bar{r}_0 = r_0$

set $z_0 = M^{-1} r_0$, $\quad P_0 = z_0$

$\bar{z}_0 = M^{-T} \bar{r}_0$, $\quad \bar{P}_0 = \bar{z}_0$

Do $i = 0,1,2,\ldots.$ *miter*

(a) Update solution

$$w_i = KP_i$$

$$\bar{w}_i = K^T \bar{P}_i$$

$$\alpha_i = \frac{\left(\bar{r}_i, z_i\right)}{\left(\bar{P}_i, w_i\right)}$$

$$x_{i+1} = x_i + \Delta x_i = x_i + \alpha_i P_i$$

$$r_{i+1} = r_i - \alpha_i w_i$$

$$\bar{r}_{i+1} = \bar{r}_i - \alpha_i \bar{w}_i$$

(b) Check convergence

If $\|r_{i+1}\|_2 < \varepsilon_1 \|b\|_2$ *or* $\|\Delta x_i\|_2 < \varepsilon_2 \|x_{i+1}\|_2$ *then stop;*
else continue

(c) Set new search direction

$$z_{i+1} = M^{-1} r_{i+1}$$

$$\bar{z}_{i+1} = M^{-T} \bar{r}_{i+1}$$

$$\beta_i = \frac{\left(\bar{r}_{i+1}, z_{i+1}\right)}{\left(\bar{r}_i, z_i\right)}$$

$$P_{i+1} = z_{i+1} + \beta_i P_i$$

$$\bar{P}_{i+1} = \bar{z}_{i+1} + \beta_i \bar{P}_i$$

EndDo;

Figure 4.9 Preconditioned Bi-CG algorithm

Bi-CG requires operations with the coefficient matrix and its transpose matrix per iteration step and terminates with n steps at most, where n is the dimension of

solution. In the case of convergence, both the residual vector $\boldsymbol{r}_{i+1}$ and $\bar{\boldsymbol{r}}_{i+1}$ converge toward zero, but only the convergence of $\boldsymbol{r}_{i+1}$ is exploited. For an un-symmetric system the method displays often a quite irregular convergence behaviour. Comparing Figure 4.9 and Figure 4.7 it is seen that nine vectors $\left(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{w}, \boldsymbol{r}, \boldsymbol{z}, \bar{\boldsymbol{P}}, \bar{\boldsymbol{w}}, \bar{\boldsymbol{r}}, \bar{\boldsymbol{z}}\right)$ and a pre-conditioner matrix $\boldsymbol{M}$ must be opened and stored in database for the Preconditioned Bi-CG algorithm, with the exception of the matrix $\boldsymbol{K}$.

In the mid-1980's Sonneveld recognized that the operations with transpose matrix $\boldsymbol{K}^T$ could be eliminated by a minor modification to the Bi-CG algorithm, without additional computational cost, which leads to the Conjugate Gradient Squared (Bi-CGS) algorithm [4.27]. In the algorithm the residual vectors $\boldsymbol{r}_j = P_j^2(\boldsymbol{K})\boldsymbol{r}_0$ are constructed, where P_j stands for the j^{th} polynomial function. By doing so, it avoided generating the vectors $\bar{\boldsymbol{r}}_j$ and doing any multiplication with the matrix $\boldsymbol{K}^T$. The Bi-CGS algorithm works quite well in many cases. However, since the polynomials are squared, rounding errors tend to be more damaging than in the Bi-CG algorithm. The Bi-Conjugate Gradient Stabilized Method (Bi-CGSTAB) evolved from Bi-CGS algorithm and it was proposed by van der Vorst in 1992 [4.29] for the solution of certain classes of non-symmetric linear systems.

In Bi-CGSTAB, instead of defining residual sequence $\boldsymbol{r}_j = \boldsymbol{P}_j^2\left(\boldsymbol{K}\right)\boldsymbol{r}_0$, the residual vectors are constructed at the j iteration step as

$$\boldsymbol{r}_j = Q_j(\boldsymbol{K})P_j(\boldsymbol{K})\boldsymbol{r}_0 \qquad 4.67$$

In which, $P_j(\boldsymbol{K})$ is a residual polynomial associated with the Bi-CG algorithm and $Q_j(\boldsymbol{K})$ is a new polynomial which is defined recursively at each iteration step with the goal of 'stabilizing' or 'smoothing' the convergence behaviour of the Bi-CG algorithm. Specifically, it can be defined by the simple recurrence

$$Q_j\left(\boldsymbol{K}\right) = (1-\Omega_{j-1})Q_{j-1}\left(\boldsymbol{K}\right) \qquad 4.68$$

where the scalar Ω_{j-1} is to be determined. The detailed derivation for the Bi-CGSTAB algorithm can be found in Reference [4.21] and the final Bi-CGSTAB algorithm for solving the linear system is presented in Figure 4.10.

(1) *Compute* $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{K}\boldsymbol{x}_0$ *for some initial guess* $\boldsymbol{x}_0$

$\bar{\boldsymbol{r}}_0 = \boldsymbol{r}_0 \quad \boldsymbol{P}_0 = 0 \quad \boldsymbol{v}_0 = 0$

$\rho_0 = 1 \quad \alpha_0 = 1 \quad \Omega_0 = 1$

(2) **Do** $j = 1, 2, \ldots.miter$

(3) $\rho_{j-1} = \left(\boldsymbol{r}_{j-1}, \bar{\boldsymbol{r}}_0\right)$

if $\left(j = 1\right)$ *then*

(4) $\boldsymbol{P}_j = \boldsymbol{r}_{j-1}$

elseif

(5) $\beta_{j-1} = \dfrac{\rho_{j-1}}{\rho_{j-2}} \dfrac{\alpha_{j-1}}{\Omega_{j-1}}$

(6) $\boldsymbol{P}_j = \boldsymbol{r}_{j-1} + \beta_{i-1}\left(\boldsymbol{P}_{j-1} - \Omega_{j-1}\boldsymbol{v}_{j-1}\right)$

endif

(7) $\boldsymbol{w}_j = \boldsymbol{M}^{-1}\boldsymbol{P}_j$

(8) $\boldsymbol{v}_j = \boldsymbol{K}\boldsymbol{w}_j$

(9) $\alpha_j = \dfrac{\rho_{j-1}}{\left(\boldsymbol{v}_j, \bar{\boldsymbol{r}}_0\right)}$

(10) $\boldsymbol{s}_j = \boldsymbol{r}_{j-1} - \alpha_j \boldsymbol{v}_j$

(11) $\boldsymbol{z}_j = \boldsymbol{M}^{-1}\boldsymbol{s}_j$

(12) $\boldsymbol{t}_j = \boldsymbol{K}\boldsymbol{z}_j$

(13) $\Omega_j = \dfrac{\left(\boldsymbol{s}_j, \boldsymbol{t}_j\right)}{\left(\boldsymbol{t}_j, \boldsymbol{t}_j\right)}$

(14) $\boldsymbol{x}_j = \boldsymbol{x}_{j-1} + \alpha_j \boldsymbol{w}_j + \Omega_j \boldsymbol{z}_j = \boldsymbol{x}_{j-1} + \Delta\boldsymbol{x}_j$

(15) $\boldsymbol{r}_j = \boldsymbol{s}_j - \Omega_j \boldsymbol{z}_j$

(16) *Check if* $\left\|\boldsymbol{r}_j\right\|_2 < \varepsilon_1 \left\|\boldsymbol{b}\right\|_2$ *or* $\left\|\Delta\boldsymbol{x}_j\right\|_2 \le \varepsilon_2 \left\|\boldsymbol{x}_j\right\|_2$ *then stop;*

(17) $\rho_{j-2} = \rho_{j-1}$

Enddo

Figure 4.10 Preconditioned Bi-CGSTAB algorithms

From Figure 4.10 it is seen that nine vectors $(\boldsymbol{x},\boldsymbol{P},\boldsymbol{r},\boldsymbol{r}_0,\boldsymbol{s},\boldsymbol{t},\boldsymbol{v},\boldsymbol{w},\boldsymbol{z})$ and pre-conditioner matrix $\boldsymbol{M}$ are created and stored in database for the Preconditioned Bi-CGSTAB algorithm, with the exception of the matrix $\boldsymbol{K}$. Compared with Bi-CG methods, Bi-CGSTAB is less expensive, more stable and converges faster. Normally we choose BiCGSTAB to solve nonlinear equations in fluid dynamics.

4.4.6 Preconditioning

As we discussed in section 4.4.3, the rate of convergence of the Krylov subspace iterative methods largely depends on the properties of the coefficients matrix in the linear system, i.e. the condition number of $\boldsymbol{K}$. In order to improve the properties of $\boldsymbol{K}$ one often transforms the linear system by a suitable linear transformation, which is termed preconditioning. Recent research is more oriented in that direction than in trying to further accelerate the Krylov subspace method. To construct an effective and efficient pre-conditioner is quite problem dependent. A pre-conditioner is considered as effective if the number of iterations of the preconditioned Krylov subspace method is reduced by the order of 100 or more. There are many different pre-conditioners proposed over the years [4.21] [4.30] and among them the incomplete Cholesky factorisation (IC) and the incomplete $\boldsymbol{LU}$ factorisation (ILU) are the most popular.

Consider a matrix $\boldsymbol{K}$ that is symmetric or un-symmetric, positive definite. Assume that the pre-conditioner matrix $\boldsymbol{M}$ is available and approximates $\boldsymbol{K}$ in some yet-undefined sense. Then the following preconditioned systems are required to be solved

$$\boldsymbol{M}^{-1}\boldsymbol{K}\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{b} \tag{4.69}$$

or

$$\boldsymbol{K}\boldsymbol{M}^{-1}\boldsymbol{u} = \boldsymbol{b}, \quad \boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{u} \tag{4.70}$$

Note that the above two systems are no longer symmetric in general, where the upper case letter $\boldsymbol{M}$ denotes a pre-conditioner. Equation (4.69) is called left preconditioning and equation (4.70) is right preconditioning. If the coefficient matrix $\boldsymbol{K}$ is a symmetric, positive definite matrix, in order to preserve the symmetry of the linear

system after transformation, the pre-conditioner $\boldsymbol{M}$ can be defined by an incomplete Cholesky factorisation (IC)

$$\boldsymbol{M} = \boldsymbol{U}^T\boldsymbol{U} \tag{4.71}$$

For example, in section 4.4.3 for preconditioned CG methods, the best choice of pre-conditioner $\boldsymbol{C}$, in order to distinguish from the general definition of $\boldsymbol{M}$, could be the inverse matrix $\boldsymbol{C} = \boldsymbol{U}^{-1}$, where $\boldsymbol{U}$ is an upper triangular matrix given by the Cholesky factorisation $\boldsymbol{K} = \boldsymbol{U}^T\boldsymbol{U}$. With this choice, the preconditioned coefficient matrix in equation (4.47) can be expressed as

$$\tilde{\boldsymbol{K}} = \boldsymbol{C}^T\boldsymbol{K}\boldsymbol{C} = \boldsymbol{U}^{-T}\boldsymbol{K}\boldsymbol{U}^{-1} = \boldsymbol{U}^{-T}\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{U}^{-1} = \boldsymbol{I} \tag{4.72}$$

The preconditioned coefficient matrix becomes equal to the identity matrix and the condition number $k = \lambda_{\max}(\boldsymbol{K})/\lambda_{\min}(\boldsymbol{K}) = 1$. This is an extreme case of a well-conditioned matrix whose eigenvalues are all 1. In practical cases, we can only construct an approximate matrix of $\boldsymbol{K}$ that can be used as a pre-conditioner. Meijerink and van der Vorst introduced a more general incomplete $\boldsymbol{LU}$ factorisation [4.31] for a symmetric or un-symmetric matrix $\boldsymbol{K}$, i.e. $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{U}$. The idea behind the ILU pre-conditioner is to modify Gaussian elimination to allow fill-in at only a restricted set of positions in the **LU** factors. Let the allowable fill-in positions be given by index set S where

$$S = \left\{ (i,j) \,\middle|\, k_{ij} \neq 0 \right\} \tag{4.73}$$

The entries of the factorised lower and upper triangular matrices are constrained by the conditions set as

$$\begin{aligned} L_{ij} &= 0 \quad if \quad j > i \quad or \quad (i,j) \notin S \\ U_{ij} &= 0 \quad if \quad i > j \quad or \quad (i,j) \notin S \end{aligned} \tag{4.74}$$

That is, the only non-zeros allowed in the **LU** factors are those for which the corresponding entries in $\boldsymbol{K}$ are non-zero. With constraints defined in equation (4.73) and (4.74) the nonzero entries of $\boldsymbol{L}$ and $\boldsymbol{U}$ can be obtained by a simple modification of the Cholesky factorisation algorithm, which is previously defined in section 4.3.2.

These incomplete $\boldsymbol{LU}$ factors are stored in the corresponding lower and upper parts of the one-dimensional array with the same dimension as $\boldsymbol{K}$, which can be used in subsequent application of the pre-conditioner. In all works of this thesis the incomplete $\boldsymbol{LU}$ factorisation (ILU) is adopted in the Krylov subspace iterative methods.

4.5 Hybrid iterative direct parallel solver

As we discussed in section 4.2.1, the implementation of the implicit parallel solver is based on the non-overlapping domain decomposition scheme; here contact problems are excluded in the implicit parallelization. The local Schur complement $\boldsymbol{S}_p$ and the condensed load vector $\boldsymbol{y}_p$ are assembled within each subdomain p, $p=1,s$, where s stands for the number of subdomains or number of corresponding slave processors. They are passed to the master processor, in which global Schur complement $\boldsymbol{S}$ and global condensed load vector $\boldsymbol{y}$ are assembled according to the numbering of interfacial nodes. The reduced system of equations, $\boldsymbol{S}\boldsymbol{x}_b = \boldsymbol{y}$, are solved by using the Krylov space iterative method, leading to a so-called hybrid iterative direct parallel solver [4.25]. The implementation of the implicit parallel solver is described in the following three parts;

(1) Master/Slave approach

(2) Blocked modified Cholesky factorisation

(3) Implicit parallel computational procedure

4.5.1 Master/Slave approach

The master/slave approach is often used in parallelization of a sequential finite element analysis code. It is a simple and effective parallel computing strategy and is adopted in this work, where the master processor serves both as a controller and worker for initiating parallel computation through notation of the slave processors and conducting the necessary sequential calculations. Such as, reading the input data from data file, performing static domain decomposition, distributing relevant loading and

supporting data, element connectivity and geometry data, material data, nodal coordinates and number list etc. into the slave processors. In the solver phase, as mentioned before, assembly of global Schur matrix and solving of the reduced system of equations are conducted by the master processor. The master processor does not perform any finite element computations. Each slave processor receives all data records from the master processor and conducts all finite element computational tasks concurrently, such as computing the global and element loading at pre-solution phase; forming the element stiffness, assembling the local Schur complement matrix and the local load vector, performing Cholesky factorisation at the solver phase; computing the element internal forces, updating nodal coordinates and outputting results on the plot files at the post-solution phase. Since each slave processor performs the same tasks, a sequential implicit code can be used with minimal modification.

4.5.2 Blocked modified Cholesky factorisation

In section 4.2.1, we already discussed the assembled local system of equations for the subdomain Ω_p, a simple procedure of static condensation, which eliminates unknown variables at internal nodes and gives the local Schur complement $\boldsymbol{S}_p$ and the effective local load vector y_p is described in equations (4.5) and (4.6). In this section, the desired matrices $\boldsymbol{S}_p$ and y_p can be found by making a slight modification to a two-blocked Cholesky factorisation, as shown in Figure 4.11.

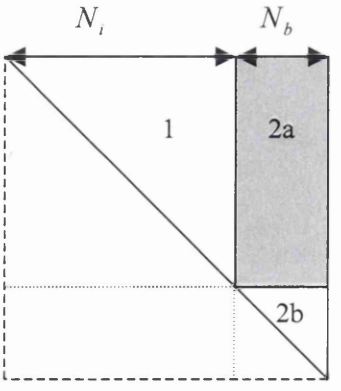


Figure 4.11 Two Blocked modified Cholesky factorization

It is assumed that the N degrees of freedom in subdomain p is divided into N_i internal d.o.f. and N_b interfacial d.o.f. The stiffness matrix in equation (4.3) can be factorised into

$$\begin{bmatrix} \boldsymbol{K}_{ii} & \boldsymbol{K}_{ib} \\ \boldsymbol{K}_{bi} & \boldsymbol{K}_{bb}^{p} \end{bmatrix} = \begin{bmatrix} \boldsymbol{L}_{ii} & \boldsymbol{0} \\ \boldsymbol{L}_{bi} & \boldsymbol{L}_{bb} \end{bmatrix} \begin{bmatrix} \boldsymbol{D}_{ii} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{D}_{bb} \end{bmatrix} \begin{bmatrix} \boldsymbol{L}_{ii}^{T} & \boldsymbol{L}_{bi}^{T} \\ \boldsymbol{0} & \boldsymbol{L}_{bb}^{T} \end{bmatrix} \tag{4.75}$$

where

$$\boldsymbol{K}_{ii} = \boldsymbol{L}_{ii}\boldsymbol{D}_{ii}\boldsymbol{L}_{ii}^{T} \tag{4.76a}$$

$$\boldsymbol{K}_{ib} = \boldsymbol{L}_{ii}\boldsymbol{D}_{ii}\boldsymbol{L}_{bi}^{T} \tag{4.76b}$$

$$\boldsymbol{K}_{bi} = \boldsymbol{L}_{bi}\boldsymbol{D}_{ii}\boldsymbol{L}_{ii}^{T} \tag{4.76c}$$

$$\boldsymbol{K}_{bb}^{p} = \boldsymbol{L}_{bi}\boldsymbol{D}_{ii}\boldsymbol{L}_{bi}^{T} + \boldsymbol{L}_{bb}\boldsymbol{D}_{bb}\boldsymbol{L}_{bb}^{T} \tag{4.76e}$$

For an un-symmetric system, $\boldsymbol{L}_{ii}^{T}$, $\boldsymbol{L}_{bi}^{T}$ and $\boldsymbol{L}_{bb}^{T}$ in equations (4.75)-(4.76) are replaced by $\boldsymbol{U}_{ii}$, $\boldsymbol{U}_{ib}$ and $\boldsymbol{U}_{bb}$, respectively. With blocked modified Cholesky factorisation of equation (4.76), the local Schur complement matrix $\boldsymbol{S}_p$ and the effective local vector $\boldsymbol{y}_p$ for a symmetric system are obtained as

$$\boldsymbol{S}_p = \boldsymbol{K}_{bb}^{p} - \boldsymbol{L}_{bi}\boldsymbol{D}_{ii}\boldsymbol{L}_{bi}^{T} \tag{4.77}$$

$$\boldsymbol{y}_p = \boldsymbol{b}_b^{p} - \boldsymbol{L}_{bi}\boldsymbol{L}_{ii}^{-1}\boldsymbol{b}_i \tag{4.78}$$

And for an un-symmetric system the local Schur complement matrix $\boldsymbol{S}_p$ is defined by

$$\boldsymbol{S}_p = \boldsymbol{K}_{bb}^{p} - \boldsymbol{L}_{bi}\boldsymbol{D}_{ii}\boldsymbol{U}_{ib} \tag{4.79}$$

The blocked modified Cholesky factorisation algorithms involve the following two steps;

Step 1: Factorizing block (1) and block (2a), it uses the modified Cholesky factorisation algorithm, which was described in section 4.3.3 as

For column $j = 2, N$

$$K_{ij} = K_{ij} - \left[\sum_{l=m}^{i-1} K_{lj}K_{li}\right]_{i>1} \qquad \begin{matrix} i = m, j-1 \quad if \quad j \le N_i \\ i = m, N_i \quad if \quad j > N_i \end{matrix} \tag{4.80}$$

$$K_{ij} = \frac{K_{ij}}{K_{ii}} \tag{4.81}$$

$$K_{jj} = K_{jj} - \sum_{l=m}^{j-1} K_{lj}^2 K_{ll} \qquad j \le N_i \tag{4.82}$$

where the term m in equation (4.80) is the starting row pointer, which is defined by equation (4.30) for a perfectly banded matrix $\boldsymbol{K}$, or equation (4.32) for a not well-banded matrix $\boldsymbol{K}$.

Step 2: The operation in this step is non-standard and it modifies block (2b) to obtain the local Schur complement matrix defined in equation (4.79)

For column $j = N_i + 1, N$

$$K_{ij} = K_{ij} - \sum_{l=m}^{N_i} K_{li} K_{lj} K_{ll} \qquad i = N_i + 1, j \qquad m = \max\left(m(j), m(i)\right) \tag{4.83}$$

The $\boldsymbol{S}_p$ matrix, which is stored in block (2b), will be copied into a one-dimensional array and sent to the master processor. In order to eliminate the internal unknown variables and obtain the effective load vector $\boldsymbol{y}_p$, equation (4.78) can be further split into two steps as

$$\boldsymbol{y}_p = \boldsymbol{b}_b^p - \boldsymbol{L}_{bi} \boldsymbol{z}_i \tag{4.84}$$

and

$$\boldsymbol{z}_i = \boldsymbol{L}_{ii}^{-1} \boldsymbol{b}_i \tag{4.85}$$

Equation (4.85) is a standard forward substitution for the internal variables and uses factors of block (1)

For $j = 2, N_i$

$$b_j = b_j - \sum_{l=m(j)}^{j-1} K_{lj} b_l \tag{4.86}$$

Equation (4.84) is a modified substitution process, which eliminates the internal variables using factors of block (2a)

For $j = N_i + 1, N$

$$b_j = b_j - \sum_{l=m(j)}^{N_i} K_{lj} b_l \qquad 4.87$$

After substitution of (4.86) and (4.87) the load vector of the subdomain p contains

$$\boldsymbol{b} = \begin{bmatrix} \boldsymbol{z}_i \\ \boldsymbol{y}_p \end{bmatrix} \qquad 4.88$$

where $\boldsymbol{z}_i = \boldsymbol{L}_{ii}^{-1}\boldsymbol{b}_i$ and $\boldsymbol{y}_p$ is the effective local load vector corresponding to interfacial unknown variables and will be sent to the master processor.

By receiving $\boldsymbol{S}_p$ and $\boldsymbol{y}_p$ from all subdomains, the final condensed equations $\boldsymbol{S}\boldsymbol{x}_b = \boldsymbol{y}$ can be solved in the master processor, then $\boldsymbol{y}_p$ in the load vector of the subdomain p is replaced by $\boldsymbol{x}_b^p$

$$\boldsymbol{b} = \begin{bmatrix} \boldsymbol{z}_i \\ \boldsymbol{x}_b^p \end{bmatrix} \qquad 4.89$$

Substituting equation (4.76) into the first equation of (4.3), the solution of internal unknown variables $\boldsymbol{x}_i$ of subdomain p is obtained by

$$\boldsymbol{x}_i = \boldsymbol{L}_{ii}^{-T}\left(\boldsymbol{D}_{ii}^{-1}\boldsymbol{z}_i - \boldsymbol{L}_{bi}^{T}\boldsymbol{x}_b^p\right) = \boldsymbol{L}_{ii}^{-T}\hat{\boldsymbol{b}}_i \qquad 4.90$$

where

$$\hat{\boldsymbol{b}}_i = \boldsymbol{D}_{ii}^{-1}\boldsymbol{z}_i - \boldsymbol{L}_{bi}^{T}\boldsymbol{x}_b^p \qquad 4.91$$

Equation (4.91) is implemented in two separate loops as

For $j = 1, N_i$ set

$$b_j = \frac{b_j}{K_{jj}} \qquad 4.92$$

For $j = N_i + 1, N$ set

$$b_l = b_l - K_{jl} b_j \qquad l = m(j), N_i \qquad 4.93$$

The backward substitution of equation (4.90) can be implemented as

For each column $j = N_i, 2$

$$b_l = b_l - K_{lj} b_j \qquad l = j-1, m(j) \qquad 4.94$$

4.5.3 Implicit parallel computational procedure

Following non-overlapping domain decomposition and blocked modified Cholesky factorisation, which are discussed in sections 4.2.1 and 4.5.2 respectively, the hybrid iterative direct parallel solution for a symmetric linear system can be summarized in Figure 4.12

(1) ***Parallel for*** $p = 1, 2, \ldots s$

(2) *Decompose* $\boldsymbol{K}_{ii}$; $\boldsymbol{K}_{ii} = \boldsymbol{L}_{ii}\boldsymbol{D}_{ii}\boldsymbol{L}_{ii}^T$

(3) $\boldsymbol{L}_{bi}^T = \boldsymbol{D}_{ii}^{-1}\boldsymbol{L}_{ii}^{-1}\boldsymbol{K}_{ib}$

(4) $\boldsymbol{z}_i = \boldsymbol{L}_{ii}^{-1}\boldsymbol{b}_i$

(5) $\boldsymbol{S}_p = \boldsymbol{K}_{bb}^p - \boldsymbol{L}_{bi}\boldsymbol{D}_{ii}\boldsymbol{L}_{bi}^T$

(6) $\boldsymbol{y}_p = \boldsymbol{b}_b^p - \boldsymbol{L}_{bi}\boldsymbol{z}_i$

(7) ***end***

(8) *Assemble global Schur complement and effective load vector*

$$\boldsymbol{S} = \sum_{p=1}^{s} \boldsymbol{S}_p$$

$$\boldsymbol{y} = \sum_{p=1}^{s} \boldsymbol{y}_p$$

(9) *Solve* $\boldsymbol{S}\boldsymbol{x}_b = \boldsymbol{y}$ *using Krylov subspace iteration*

(10) ***Parallel for*** $p = 1, 2, \ldots s$

(11) calculate internal variables

$$\boldsymbol{x}_i = \boldsymbol{L}_{ii}^{-T}(\boldsymbol{D}_{ii}^{-1}\boldsymbol{z}_i - \boldsymbol{L}_{bi}^T\boldsymbol{x}_b^p)$$

(12) ***end***

Figure 4.12 A hybrid iterative direct parallel solution for a linear system

In Figure 4.12 steps (2)-(7) involve Cholesky factorisation and eliminate the unknown variables of internal nodes at each subdomain. Data sent back to the master processor are $\boldsymbol{S}_p$ and $\boldsymbol{y}_p$. Assembly of the global Schur matrix and the effective load vector is

undertaken in step(8). In step (9) the reduced system of equations is solved for the interfacial variables x_b. Direct solution of the reduced system involves the dominating part of the total computational costs, and this is where we employ the Krylov subspace iterative algorithm, ICCG, GMRES, and Bi-CGSTAB. After solving x_b the data sent to each slave processor is x_b^p, the variables at interfacial nodes related with subdomain p. The calculation of unknown variables at internal nodes x_i is done in parallel by steps (10)-(12).

For a nonlinear system, steps (1)-(12) in Figure 4.12 are nested in the iteration, as shown in Figure 4.13.

(1) *Do itera=1,mtera*

(2) ***Parallel for*** *p=1,2...,s*

.....

end

Solve a reduce system

(3) ***Parallel for*** *p=1,2...,s*

(4) $x_i = L_{ii}^{-T}(D_{ii}^{-1}z_i - L_{bi}^T x_b^p)$

(5) calculate internal forcess

$$\bar{f}_p = \begin{Bmatrix} f_i \\ f_b^p \end{Bmatrix} = \int_{\Omega_p} B\sigma d\Omega$$

(6) ***end***

(7) $f = \sum_{p=1}^{S} \bar{f}_p$

(8) *if(itera=1)* $b = \sum_{p=1}^{S} \bar{b}_p = \sum_{p=1}^{S} \begin{Bmatrix} b_i \\ b_b^p \end{Bmatrix}$

(9) *check convergence*

$\|f - b\|_2 \le \varepsilon_1 \|b\|_2$ *and* $\|\Delta x\|_2 \le \varepsilon_2 \|x\|_2$ *stop*

(10) *enddo*

Figure 4.13 A hybrid iterative direct parallel solution for a nonlinear system

In Figure 4.13 the calculation of internal forces $\overline{f}_p$ at each slave processor is defined by step (5) in parallel, where f_i and f_b^p are the internal forces related with internal nodes and interfacial nodes of subdomain p, respectively. The convergence check has to be made within the whole domain in the master processor at step (9); the sudomain's incremental displacement, internal force and external loading vectors $\Delta\overline{x}_p$, $\overline{f}_p$ and $\overline{b}_p$ are sent back to the master processor. If the residual norm and displacement norm are less than defined tolerances, it assumes that convergence is reached and the non-linear system is solved completely. It is noted that the global external force vector is only assembled at the first iteration of each increment in step (8).

4.6 Explicit parallel solver for fluid dynamics

The parallelization of explicit finite element fluid dynamics is based on the Schwarz alternating procedure. Since the main computational steps of the explicit fluid dynamics solution involve internal force calculation of the Lagrangian fluid elements, the contact detection and contact force calculation at the fluid-structure boundary or interaction forces between fluid particles, it is necessary to use an overlapping domain decomposition scheme, as described in section 4.2.2. The literature dealing specially with overlapping partitioning is not extensive; we are aware of publications dealing with a similar subject in fluid dynamics by Farhat and Lanteri [4.33][4.34] and in solid dynamics by Krysl and Bittnar [4.8]. Other examples of using an overlapping domain decomposition method for explicit finite/discrete element dynamic analysis is by Owen and Feng *et al* [4.11][4.35]. The implementation of the explicit parallel solver is described in detail by the following three parts:

(1) Classification of element and nodes
(2) Time integration of governing equations
(3) Explicit parallel computation procedure

4.6.1 Classification of element and nodes

The overlapping domain decomposition scheme shown in Figure 4.2 can be slightly modified and changed to a simple axis-aligned automatic partitioning algorithm, in which each subdomain is confined to a rectangular box in 2-D problem or a parallelepiped box in 3-D problem [4.11][4.35]. A buffer zone between the adjacent subdomains is introduced, half of the zone which is located inside of the subdomain boundary is called an interfacial zone, the other half outside of the boundary is defined as an external zone. The buffer size should be larger than the maximum size of elements located in the buffer zone. Therefore, each subdomain is divided into three zones; internal, interfacial and external zone, as shown in Figure 4.14

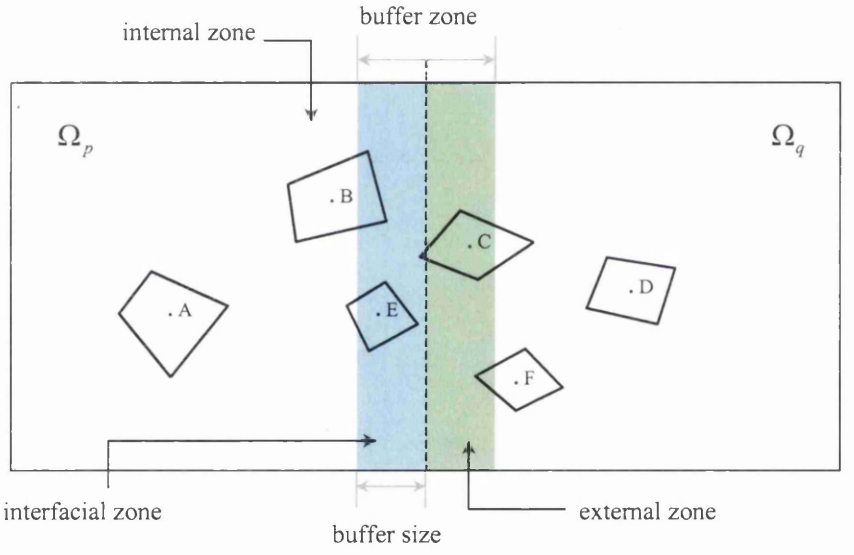


Figure 4.14 Classification of elements and nodes

In Figure 4.14, the classification of continuum or discrete elements is made according to the location of the centre point of the element. The centre point of any element inside of the internal zone is defined as an internal element, such as element A and B in subdomain Ω_p. If the centre point of an element is inside the interfacial zone, it is defined as an interfacial element, such as element E. Other elements with their centre on the external zone are defined as an external element, such as element C in subdomain Ω_p. In this work element migration across subdomain boundary is not

considered, the purpose of classification of elements is to set nodal definition. The classification of element nodes is the same as element definition according to their spatial position, but there are two special situations, which must be treated carefully. If a node of an external element of subdomain Ω_p is located inside the internal zone of the adjacent subdomain Ω_q, it is still called an external node. A node of an interfacial element located inside of the internal zone is defined as an interfacial node. Two nodal list arrays, which store interfacial and external nodal pointers in the global nodal list, are created for each subdomain's boundary for handling the inter-processor communication and updating unknown variables at external nodes during the Schwarz alternating procedure.

4.6.2 Time integration of the governing equations

The explicit Euler formulations of the finite element discretization for transient Stokes flow are assumed balanced at time step n and given by equation (2.65), which can be rewritten as

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + \Delta t \boldsymbol{M}^{-1}\left[{}^*\boldsymbol{F}_u - \left({}^*\boldsymbol{K}\boldsymbol{u}_n + {}^*\boldsymbol{Q}\boldsymbol{P}_n\right)\right] = \boldsymbol{u}_n + \Delta t \boldsymbol{M}^{-1}\left({}^*\boldsymbol{F}_u - \boldsymbol{F}_u^{\text{int}}\right) \quad 4.95$$

$$\boldsymbol{P}_{n+1} = \boldsymbol{P}_n + \Delta t \boldsymbol{M}_\rho^{-1}\left[-{}^*\boldsymbol{F}_P - \left(-{}^*\boldsymbol{Q}^T\boldsymbol{u}_n + {}^*\boldsymbol{M}_P\boldsymbol{P}_n\right)\right] = \boldsymbol{P}_n + \Delta t \boldsymbol{M}_\rho^{-1}\left(-{}^*\boldsymbol{F}_P - \boldsymbol{F}_P^{\text{int}}\right) \quad 4.96$$

In other words the velocity $\boldsymbol{u}_{n+1}$ and pressure $\boldsymbol{P}_{n+1}$ at time step $n+1$ are given explicitly in terms of the velocity $\boldsymbol{u}_n$ and pressure $\boldsymbol{P}_n$ at time step n. The definition of other terms can be referred to in section 2.8. Since the mass matrix $\boldsymbol{M}$ and $\boldsymbol{M}_\rho$ are diagonal, then the solution of (4.95) and (4.96) becomes trivial and can be given by

$$\left(u_i\right)_{n+1} = \left(u_n\right)_i + \Delta t m_{ii}^{-1}\left[\left({}^*F_{u,i}\right)_n - \left(F_{u,i}^{\text{int}}\right)_n\right] \quad 4.97$$

$$\left(P_i\right)_{n+1} = \left(P_i\right)_i + \Delta t m_{\rho,ii}^{-1}\left[-\left({}^*F_{p,i}\right)_n - \left(F_{p,i}^{\text{int}}\right)_n\right] \quad 4.98$$

In which, the right subscript i denotes the i^{th} degree of freedom. $\left({}^{*}F_{u,i}\right)_n, \left(F_{u,i}^{\text{int}}\right)_n$ are the i^{th} d.o.f. of the applied nodal force and internal force at time step n corresponding to the momentum equations. $\left({}^{*}F_{p,i}\right)_n, \left(F_{p,i}^{\text{int}}\right)_n$ are the i^{th} d.o.f. of the applied nodal force and internal force at time step n related with the continuum equations. The displacement $\left(d_i\right)_{n+1}$ and the coordinates of the Lagragian mesh $\left(x_i\right)_{n+1}$ can be simply updated by

$$\left(d_i\right)_{n+1} = \left(d_i\right)_n + \Delta t \left(u_i\right)_{n+1} \qquad 4.99$$

$$\left(x_i\right)_{n+1} = \left(x_i\right)_n + \Delta t \left(u_i\right)_{n+1} \qquad 4.100$$

4.6.3 Explicit parallel computational procedure

The explicit parallel computational procedure for a fluid dynamics system can be summarized and illustrated in the program flow chart of Figure 4.15. It can be seen that the inter-processor communication is reduced to a minimum, only the nodal velocity and pressure at external nodes for each subdomain need to be replaced by those at interfacial nodes of the adjacent subdomains at each time step. The inter-processor communication is limited to the adjacent subdomains, which share the same boundary. After receiving the data from the master processor, the work carried out by the slave processors is explained as follows:

Step (1) Initialise data; in which a solution order for the global equations is established. A global table record is created; it includes lists of global nodal pointers, equation numbers and number of degrees of freedom for each active nodes. A solution record is created to store the nodal variables results, nodal mass, nodal loads caused by element and global loading. In each element group, element processing and results records are opened in the database. A list of the element nodal pointers, which point to the positions of the global nodal pointers, is created in the element processing record. With the aid of those nodal pointers it can retrieve nodal velocities at the element level and assemble element internal forces into the global force vector.

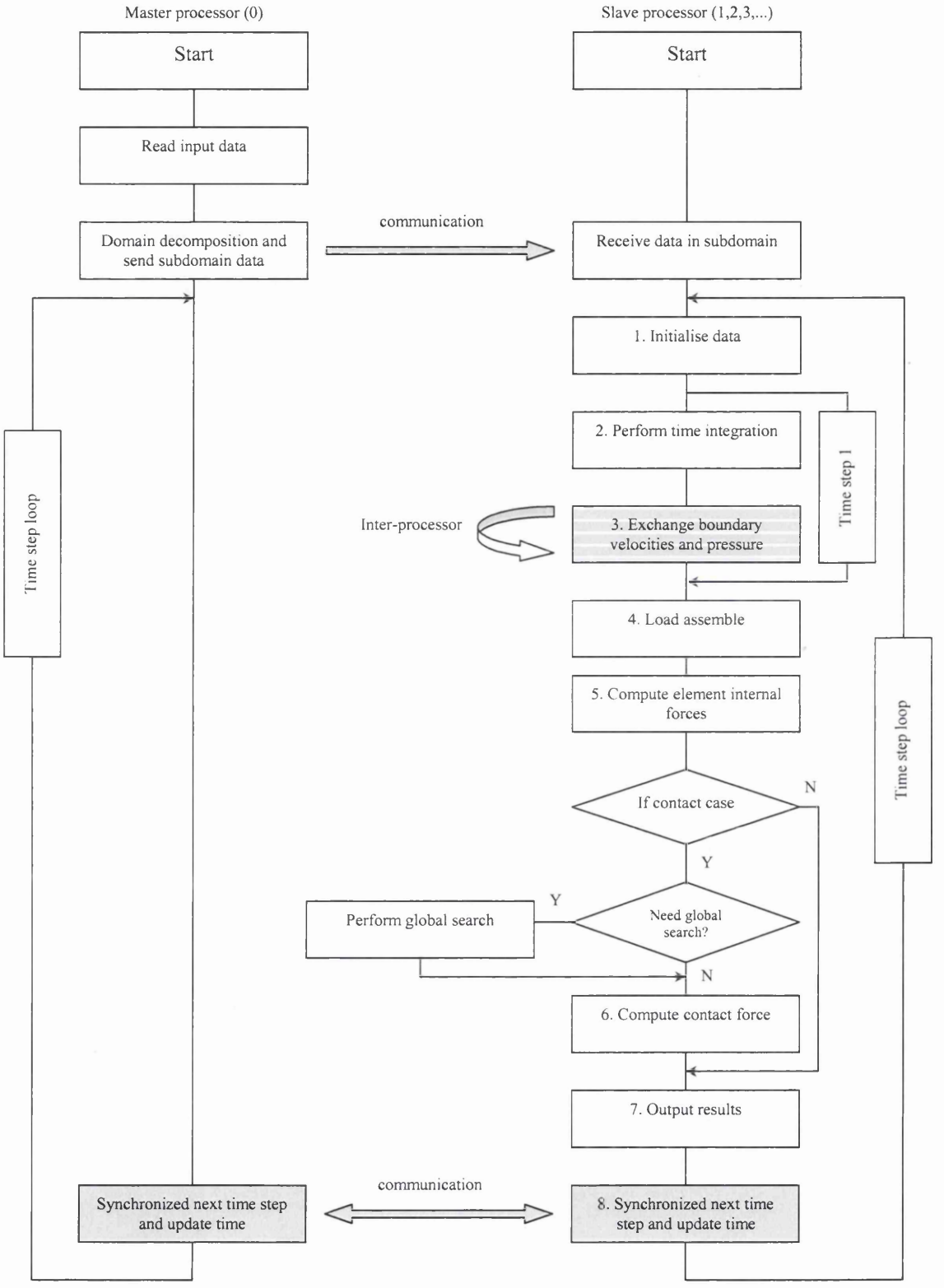


Figure 4.15 Program flow chart of the explicit parallel computational procedure

Step (2) Perform time integration; the nodal velocities and pressure are integrated according to equation (4.97) and (4.98), nodal displacements and coordinates are updated at the same time.

Step (3) Exchange boundary velocities and pressure; the nodal velocities and pressure at external nodes for each subdomain are replaced by those at interfacial nodes of the adjacent subdomains.

Step (4) Assemble external forces; the external forces consist of the global nodal loading and element loading, which includes element face loading and gravity loading.

Step (5) Calculate element internal forces; the element internal forces are evaluated according to equations (4.95) and (4.96) for transient Stokes flow. At the same time the lumped element nodal mass defined in equations (2.92) and (2.93) is assembled into the global mass vector.

Step (6) Calculate nodal contact forces; calculation of discrete element contact forces comprises two major parts, contact detection and computation of the contact forces using node-to-facet 2D or 3D formulations as defined in section 3.1. Generally, contact force calculation consumes over 40% of the execution time in sequential analysis. Since the number of contact points is dramatically reduced in each subdomain, it effectively reduces the cost of generating the binary tree. In addition, the global search is only carried out when the maximum nodal displacement in a subdomain is larger than a pre-defined value.

Step (7) Output results; each processor generates its own results and plotting files.

Step (8) Synchronize the next time step and update time; each processor defines a critical time Δt_{cr} within its own internal and interfacial elements only, then the analysis system synchronizes to obtain a minimum time step as the next time step for all processors.

4.7 Appendix

4.7.1 Modified Gram-Schmidt orthonormalizing process

A set of vectors $\boldsymbol{G}=\{\boldsymbol{a}_1,\boldsymbol{a}_2,\ldots\boldsymbol{a}_k\}$ is said to be orthonormal if

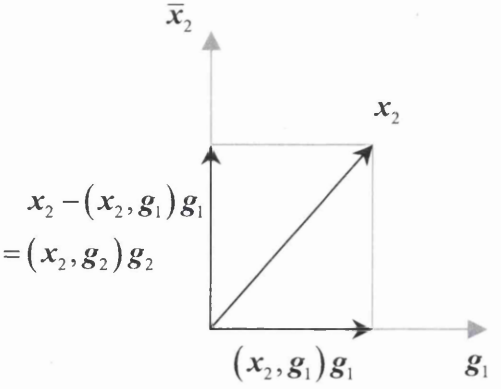
$$\begin{aligned}(\boldsymbol{a}_i,\boldsymbol{a}_j)&=0 \qquad if\ i\neq j\\ (\boldsymbol{a}_i,\boldsymbol{a}_j)&=1 \qquad if\ i=j\end{aligned} \tag{A4.1}$$

where $(\ ,\ )$ denotes vector inner products. Given a set of linearly independent vectors $\boldsymbol{X}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots\boldsymbol{x}_r\}$ and $\boldsymbol{X}$ is a $n\times r$ dimensional matrix. There are several methods to orthonormalize the vectors in $\boldsymbol{X}$. The Gram-Schmidt algorithm and Householder algorithm are two important orthonormalizing processes. The standard Gram-Schmidt process can be described as follows.

(1) Firstly normalise $\boldsymbol{x}_1$

$$\boldsymbol{g}_1=\frac{\boldsymbol{x}_1}{\|\boldsymbol{x}_1\|_2} \tag{A4.2}$$

(2) Assume new direction $\bar{\boldsymbol{x}}_2\perp\boldsymbol{g}_1$



$$\bar{\boldsymbol{x}}_2=\boldsymbol{x}_2-(\boldsymbol{x}_2,\boldsymbol{g}_1)\boldsymbol{g}_1$$

$$\boldsymbol{g}_2=\frac{\bar{\boldsymbol{x}}_2}{\|\bar{\boldsymbol{x}}_2\|_2} \tag{A4.3}$$

(3) The J^{th} step of the Gram-Schmidt process consists of orthogonalizing the vector $\boldsymbol{x}_j$ against all previous vectors $\boldsymbol{g}_{j-1}$

$$\bar{x}_j = x_j - \sum_{i=1}^{j-1}(x_j, g_i)g_i = x_j - \sum_{i=1}^{j-1} r_{i,j} g_i$$

$$g_j = \frac{\bar{x}_j}{\|\bar{x}_j\|_2} \quad \text{and} \quad r_{i,j} = (x_j, g_i) \qquad \text{A4.4}$$

If and only if a set of vectors $\{x_1, x_2, \ldots x_r\}$ is linearly independent, then a set of orthonormalized vectors $\{g_1, g_2, \ldots g_r\}$ will be completed at r steps without breaking down. The modified Gram-Schmidt algorithm (MGS) has better numerical properties than the standard Gram-Schmidt procedure, and is given in Figure 4.16

1) Define $g_1 = \frac{x_1}{\|x_1\|_2}$

Do $j = 2, \ldots r$

2) Set $\hat{g} = x_j$

Do $i = 1, \ldots j-1$

3) $r_{i,j} = (\hat{g}, g_i)$

4) $\hat{g} = \hat{g} - r_{i,j} g_i$

EndDo

5) Compute $r_{j,j} = \|\hat{g}\|_2$

6) **If** $(r_{j,j} = 0)$ then

Stop

Else

$g_j = \frac{\hat{g}}{r_{j,j}}$

EndIf

EndDo

Figure 4.16 Modified Gram-Schmidt algorithm

4.7.2 Arnoldi's process

Arnoldi's process is an algorithm that is very similar to the Modified Gram-Schmidt method but used for building an orthogonal basis of the Krylov subspace κ_m. Given a set of vectors $\{v_1, v_2, \ldots v_m\}$ in subspace κ_m, each vector v_j is of the form $q_{j-1}(K)v_1$

where q_{j-1} is a polynomial of degree $j-1$. With the modified Gram-Schmidt process, the Arnoldi algorithm takes the following form, as shown in Figure 4.17

1) Define $v_1 = \frac{v_1}{\|v_1\|_2}$

Do $j = 1, 2, ...m$

2) Compute $w_j = \boldsymbol{K} v_j$

Do $i = 1, \ldots j$

3) $h_{i,j} = (w_j, v_i)$

4) $w_j = w_j - h_{i,j} v_i$

EndDo

5) Compute $h_{j+1,j} = \|w_j\|_2$

6) **If** ($h_{j+1,j} = 0$) then

Stop

Else

$$v_{j+1} = \frac{w_j}{h_{j+1,j}}$$

EndIf

EndDo

Figure 4.17 Arnoldi - Modified Gram-Schmidt algorithm

An important proposition is given without proof, detailed proof can be found in Reference [4.21].

Proposition

Assuming that Arnoldi's process does not stop before m-step, then the vectors $\{v_1, v_2, \ldots v_m\}$ form an orthonormal basis of the Krylov space

$$\kappa_m = span\{v_1, \boldsymbol{K} v_1, \ldots \boldsymbol{K}^{m-1} v_1\} \qquad \text{A4.5}$$

4.7.3 QR algorithm

In order to solve the least-square problem $\min\left\|\beta e_1 - \bar{H}_m y_m\right\|_2$ it is natural to transform the Hessenberg matrix into an upper triangular form by using plane rotations. The rotation matrices $\Omega_i, i=1,m$ can be defined as

$$\Omega_i = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & c_i & s_i & & \leftarrow row\ i \\ & & & -s_i & c_i & & \leftarrow row\ i+1 \\ & & & & & 1 & \\ & & & & & & \ddots \\ & & & & & & 1 \end{bmatrix} \qquad \text{A4.6}$$

where $c_i^2 + s_i^2 = 1$. If m iterations of the GMRES steps (3)-(20) in Figure 4.8 are performed, then Ω_i has a dimension of $(m+1)\times(m+1)$.

Left-multiply the Hessenberg matrix $\bar{H}_m$ and the corresponding right-hand $\hat{b}_0 = \beta e_1$ by a sequence of rotation matrices, then the coefficients of s_i, c_i can be selected to eliminate $h_{i+1,i}$ at each time. For example, if $m=5$, the Hessenberg $\bar{H}_5$ and $\hat{b}_0$ are given by

$$\bar{H}_5 = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} \\ & h_{32} & h_{33} & h_{34} & h_{35} \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{bmatrix} \qquad \hat{b}_0 = \begin{bmatrix} \hat{b}_1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \beta \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \text{A4.7}$$

Then left-multiply matrix $\bar{H}_5$ and $\hat{b}_0$ by Ω_1, which is set as

$$\Omega_1 = \begin{bmatrix} c_1 & s_1 & & & & \\ -s_1 & c_1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix} \qquad \text{A4.8}$$

with

$$s_1 = \frac{h_{21}}{\sqrt{h_{11}^2 + h_{21}^2}} \qquad c_1 = \frac{h_{11}}{\sqrt{h_{11}^2 + h_{21}^2}} \tag{A4.9}$$

It results in the matrix and right-hand side vector being

$$\bar{\boldsymbol{H}}_5^{(1)} = \begin{bmatrix} h_{11}^{(1)} & h_{12}^{(1)} & h_{13}^{(1)} & h_{14}^{(1)} & h_{15}^{(1)} \\ & h_{22}^{(1)} & h_{23}^{(1)} & h_{24}^{(1)} & h_{25}^{(1)} \\ & h_{32} & h_{33} & h_{34} & h_{35} \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{bmatrix} \quad \hat{\boldsymbol{b}}_1 = \begin{bmatrix} \hat{b}_1 \\ \hat{b}_2 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} c_1\beta \\ -s_1\beta \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{A4.10}$$

The above matrix and right-hand side vector are again left multiplied by a rotation matrix Ω_2 to eliminate h_{32} with

$$s_2 = \frac{h_{32}}{\sqrt{\left(h_{22}^{(1)}\right)^2 + h_{32}^2}} \qquad c_2 = \frac{h_{22}^{(1)}}{\sqrt{\left(h_{22}^{(1)}\right)^2 + h_{32}^2}} \tag{A4.11}$$

This elimination process is continued until the m^{th} rotation is applied, which transforms the problem into an upper triangular matrix and right-hand side vector as

$$\bar{\boldsymbol{H}}_5^{(5)} = \begin{bmatrix} h_{11}^{(5)} & h_{12}^{(5)} & h_{13}^{(5)} & h_{14}^{(5)} & h_{15}^{(5)} \\ & h_{22}^{(5)} & h_{23}^{(5)} & h_{24}^{(5)} & h_{25}^{(5)} \\ & & h_{33}^{(5)} & h_{34}^{(5)} & h_{35}^{(5)} \\ & & & h_{44}^{(5)} & h_{45}^{(5)} \\ & & & & h_{55}^{(5)} \\ & & & & 0 \end{bmatrix} \quad \hat{\boldsymbol{b}}_5 = \begin{bmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \cdot \\ \cdot \\ \cdot \\ \hat{b}_6 \end{bmatrix} \tag{A4.12}$$

where the elements c_i and s_i of the *ith* rotation Ω_i are defined as.

$$s_i = \frac{h_{i+1,i}}{\sqrt{\left(h_{i,i}^{(i-1)}\right)^2 + h_{i+1,i}^2}} \quad c_i = \frac{h_{i,i}^{(i-1)}}{\sqrt{\left(h_{i,i}^{(i-1)}\right)^2 + h_{i+1,i}^2}} \tag{A4.13}$$

Define $\boldsymbol{Q}_m$ as the product of Ω_i

$$\boldsymbol{Q}_m = \Omega_m \Omega_{m-1} \ldots \Omega_1 \qquad \text{A4.14}$$

and

$$\bar{\boldsymbol{R}}_m = \bar{\boldsymbol{H}}_m^{(m)} = \boldsymbol{Q}_m \bar{\boldsymbol{H}}_m \qquad \text{A4.15}$$

$$\hat{\boldsymbol{b}}_m = \boldsymbol{Q}_m \beta \boldsymbol{e}_1 = \left(\hat{b}_1, \ldots \hat{b}_{m+1}\right)^T \qquad \text{A4.16}$$

Since $\boldsymbol{Q}_m$ is unitary, the above least-square problem $\min \left\| \beta \boldsymbol{e}_1 - \bar{\boldsymbol{H}}_m \boldsymbol{y}_m \right\|_2$ is equivalent to

$$\min \left\| \beta \boldsymbol{e}_1 - \bar{\boldsymbol{H}}_m \boldsymbol{y}_m \right\|_2 = \min \left\| \hat{\boldsymbol{b}}_m - \bar{\boldsymbol{R}}_m \boldsymbol{y}_m \right\|_2 \qquad \text{A4.17}$$

and

$$\bar{\boldsymbol{R}}_m \boldsymbol{y}_m = \hat{\boldsymbol{b}}_m \qquad \text{A4.18}$$

The solution to the above least-square problem is simply obtained by solving the triangular system (A4.18) resulting from deleting the last row of the matrix $\bar{\boldsymbol{R}}_m$ and $\hat{\boldsymbol{b}}_m$. In addition the residual norm of $\left\| \beta \boldsymbol{e}_1 - \bar{\boldsymbol{H}}_m \boldsymbol{y}_m \right\|_2$ is equal to $\left| \hat{b}_{m+1} \right|$ in $\hat{\boldsymbol{b}}_m$ of equation (A4.16).

4.8 References

[4.1] C. Farhat and E. Wilson, A new finite element concurrency computer program architecture, *Int. J. Numer. Meth. Engng.* Vol 24 (1987) 1771-1792

[4.2] B. Nour-Omid, A. Raefsky and G. Lyzenga, Solving finite element equations on concurrent computers, in A. K. Noor (ed.) *Parallel computations and their impact on mechanics,* ASME. New York. (1987) 209-228.

[4.3] A.E. Elwi and D. W. Murray, Skyline algorithms for multilevel substucture analysis, *Int. J. Numer. Meth. Engng.* Vol 21 (1985) 465-479.

[4.4] G. Fen Paw and D.R.J. Owen, Domain decomposition procedures for finite element analysis on transputer based parallel computers, *Comput. Sys. Engng.* Vol. 2 (1991) 451-460

[4.5] H.A. van der Vorst, Efficient and reliable iterative methods for linear systems, *J. Comput. Appl. Math.* 149 (2002) 251-265.

[4.6] C. Farhat, F.-X. Roux, A method of finite element tearing and interconnecting and its parallel solution algorithm, *Int. J. Numer. Meth. Engng.* 32 (1991) 1205-1227

[4.7] C. Farhat, K. Pierson, M. Lesoinne, The second generation FETI methods and their application to the parallel solution of large-scale linear and geometrically nonlinear structural analysis problems, *Comput. Methods Appl. Mech. Engrg.* 184, (2000) 333-374

[4.8] P. Krysl and Z. Bittnar, Parallel explicit finite element solid dynamics with domain decomposition and message passing: dual partitioning scalability, *Computer and Structures* Vol. 79, (2001) 345-360.

[4.9] K.T. Danielson and R.R. Namburu, Nonlinear dynamic finite element analysis on parallel computers using Fortran 90 and MPI, *Adv Engng. Soft.* 29 (1998) 179-186

[4.10] S. Plimpton, S. Attaway, B. Henrickson, J. Swegle, C. Vaughan, D. gardner, Parallel transient dynamics simulations: algorithms for contact detection and smoothed particle hydrodynamics, *J. Parallel. Distrib. Comput.* 50 (1998) 104-122.

[4.11] D.R.Owen, Y.T.Feng, E.A. de Souza Neto, M.G. Cottrell, F. Wang, F.M. Andrade Pires and J. Wu, The modelling of multi-fracturing solids and particulate media. *Int. J. Numer. Meth. Engng.* Vol. 60, (2004), 317-339.

[4.12] William Group, Ewing Lusk, and Anthony Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface. The MIT Press, Massachusetts (1994)

[4.13] William Group, Ewing Lusk, and Rajjev Thakur, Using MPI-2:Advanced Features of the Message-Passing Interface. The MIT Press, Cambridge, Massachusetts (1999)

[4.14] M.A. Crisfield, Finite element and solution procedures for structural analysis Vol 1: Linear analysis. Pineridge, Swansea (1986)

[4.15] M.R Hestens, E.L Stiefel, Methods of conjugate gradients for solving linear systems, *J. Res. Nat. Bur. Standards.* Section B 49 (1952) 409-436.

[4.16] C. Lanczos, Solution of systems of linear equations by minimized iterations, *J. Res. Nat. Bur. Standards.* 49 (1952) 33-53.

[4.17] J.W. Daniel, The conjugate gradient method for linear and nonlinear operator equations, SIAM. *J. Numer. Anal.* 4 (1967) 10-26.

[4.18] J.K. Reid, On the method of conjugate gradients for the solution of large sparse systems of linear equations, in: J.K. Reid (Ed.), Large Sparse Sets of Linear equation, Academic Press, New York, (1971) 231-254.

[4.19] J.K. Reid, The use of conjugate gradients for systems of equations possessing 'Property A', SIAM *J. Numer. Anal.* (1972) 325-332.

[4.20] D.S. Kershaw, The incomplete Choleski-conjugate gradient method for the iterative solution of systems of linear equations, *J. Comput. Phys.* 26 (1978) 43-65.

[4.21] Y. Saad, Iterative methods for sparse linear systems, 2nd edition. SIAM, 2003.

[4.22] U. Kirsch, M. Kocvara and J.Zowe, Accurate reanalysis of structures by a preconditioned conjugate gradient method. *Int. J. Numer. Meth. Engng.*, Vol. 55, (2002) 233-251.

[4.23] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM *J. Sci. Statist. Comput.* 7 (1986) 856-869.

[4.24] W.E. Arnoldi, The principle of minimized iteration in the solution of the matrix eigenvalue problem, *Quart. Appl. Math.* 9 (1951) 17-29.

[4.25] H.A. van der Vorst, Efficient and reliable iterative methods for linear systems, *J. Comput. Appl. Math.* 149 (2002) 251-265.

[4.26] J. Higgins, Introducing GMRES: a simple analysis of the algorithm, Department of Mathematics, South Dakota School of Mines and Technology, (2004)

[4.27] P. Sonneveld, CGS: a fast Lanczos-type solver for nonsymmetric linear systems, SIAM *J. Sci. Statist. Comput* 10 (1989) 36-52.

[4.28] R.W. Freund and N.M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear system. *Num. Math.* 60 (1991) 315-339.

[4.29] H.A. van der Vorst, Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems. SIAM *J. Sci. Statist. Comput.* 13 (1992) 631-644.

[4.30] Y. Saad and H.A. van der Vorst, Iterative solution for linear systems in the 20^{th} century, *J. Comput. Appl. Math.* 123 (2000) 1-33.

[4.31] J.A.Meijerink, H.A. van der Vorst, Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems, *J. Comput. Phys.* 44 (1981) 134-155.

[4.32] V. Faber, T.A. Manteuffel, Necessary and sufficient conditions for the existence of a conjugate gradient method, SIAM *J. Numer. Anal.* 21 (2) (1984) 352-362.

[4.33] CH. Farhat and S. Lanteri, Simulation of compressible viscous flow on a variety of MPPs: computational algorithm for unstructured dynamic meshes and performance results. *Technical Report No. 2154, INRIA*, January (1994).

[4.34] S. Lanteri, Parallel solutions of compressible flows using overlapping and non-overlapping mesh partitioning strategies. *Parallel Comput.*, Vol. 22, (1996) 943-968.

[4.35] D.R.Owen and Y.T.Feng, Parallelised finite/discrete element simulation of multi-fracturing solids and discrete systems. *Engineering Computations*, 18 (3/4): (2001), 557-576.

Chapter 5

Numerical Examples

In order to validate and assess the performance and implementation of the finite element formulations discussed in the previous chapters, a number of numerical simulations are presented and compared with relevant experimental test cases in the following sections.

5.1 Sloshing water waves

The importance of the stability of a vehicle or vessel transporting liquids in a tank has led to a need for understanding the strong and violent liquid motion inside containers. The violent impacts induce very large peak pressure on the tank wall, which may cause instability of the vehicle during the transportation. The sloshing effect in the ballast tank of a ship may experience large rotation motion affecting stability of the ship. Therefore, the numerical simulation of wave sloshing, which can accurately predict the free surface motion of the waves, is highly useful. The developments of numerical methods modelling steep or overturning waves are reported by several authors [5.1][5.2]. Turnbull *et al.* [5.1] simulated 2-D forced sloshing in a horizontally accelerated tank filled with inviscid liquids with a finite element analysis scheme. More recently, Bredmose *et al.* [5.2] reported experimental observations of free-surface waves caused by harmonic forced accelerations and used an extended set of Boussinesq equations [5.3] to model standing waves. In the test, the space-time Galerkin/least-squares finite element formulation for a slightly compressible transient

Stokes flow, which was developed in Chapter 2, are adopted to simulate the sloshing water waves. The experimental data and results are obtained from Bredmose's works.

5.1.1 Experimental set up

The experiments of water sloshing in a glass tank were conducted at the Civil Engineering Department of Bristol University. A rectangular, narrow glass water tank with dimension L1480×W400×H750 mm was fixed on a shaking table and subjected to a horizontal excitation in a direction parallel to the long side of the glass tank, shown in Figure 5.1. The whole experiments were recorded by two stationary video cameras: one regular-speed camera (25 frames per second) and one high-speed camera (200 frames per second). All measurements and comparisons are taken on the front face of the tank. The experimental results (high-speed photography) are of sufficient quality and quantity to allow comparison and verification of the simulations. The numerical results are compared to snapshots taken by those cameras at successive times.

The object of the validation is to demonstrate that the finite element fluid analysis can correctly and accurately depict wave propagation within the tank that is excited by a prescribed motion. Therefore, the horizontal acceleration of the tank is carefully recorded in Figure 5.2, which illustrates the build up of possible wave sloshing at time t = 5.9 second, starting from a sinusoidal oscillation. Then the tank is pushed forcefully toward the left, which produces the strong negative peak, followed by a positive acceleration.

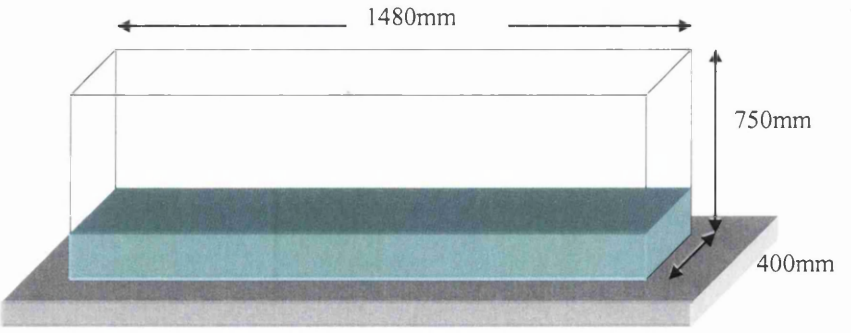


Figure 5.1. Sketch of the experimental setup

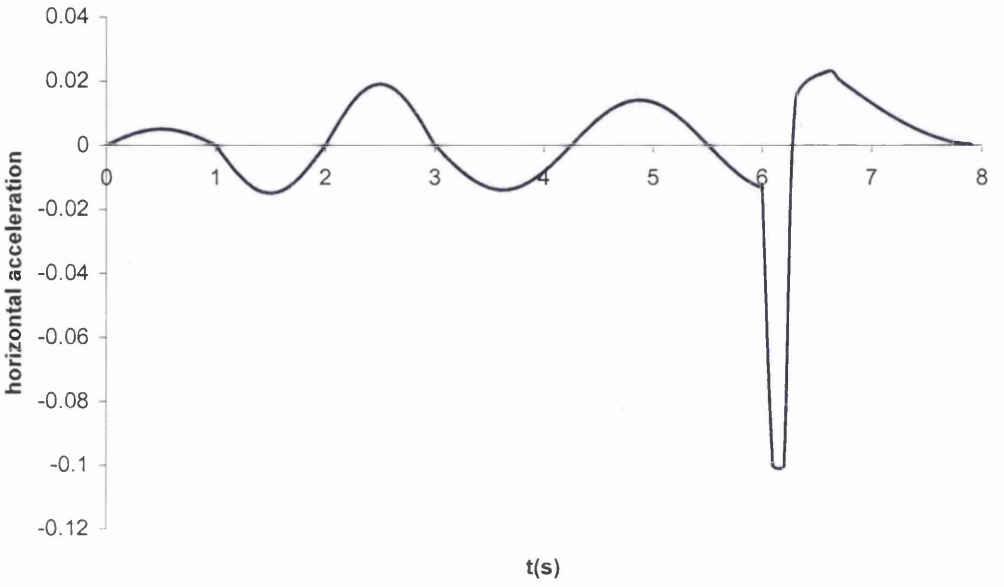


Figure 5.2. A horizontal tank acceleration record

5.1.2 2-D implicit finite element modelling of horizontal water sloshing

The 2-D implicit simulation of water sloshing is set as a plane strain problem. The water depth in the tank is 155 mm high, which is corresponding to the experiment test case labelled H10 [5.2]. The properties of the water are listed in Table 5.1.

Property	***Value***
Bulk modulus	$K = 2.15\times10^{3}\ N/mm^{2}$
Density	$\rho = 1.0\times10^{-9}\ N\,\text{sec.}^{2}/mm^{4}$
Viscosity	$\mu = 1.0\times10^{-9}\ N\,\text{sec.}/mm^{2}$

Table 5.1 Material properties

The liquid is initially under gravity loading, the gravitation g is set to be $9800\,mm/\text{sec}^{2}$. The initial finite element mesh contains 1997 linear triangular

elements and 1107 nodes, as shown in Figure 5.3a. The corresponding boundary conditions are set in Figure 5.3b, where the prescribed x-direction velocity $\overline{u}$ is given through the integration of the acceleration record and zero pressure is defined on top of free surface. The stabilisation constant, which is defined in equation 2.33, is set as $\alpha = 0.2\times10^3$. In order to compare with the snapshot of experimental results, the time step is chosen as $\Delta t = 0.01$ second and the plotting files are output at every 0.02 sec. of the time interval. The total simulation time is 8.0 second. The mesh is checked in every 5 steps with an allowable distortion angle of 5 degrees, the maximum and minimum allowable angles are 165 and 15 degrees, respectively. At each mesh adaptive stage the error estimator based on the velocity gradients is used to predict the size of elements in a new mesh. The maximum and minimum size of elements on the adaptive new mesh are set as $l_{max} = 30mm$ and $l_{min} = 15mm$. Figure 5.3c gives the initial pressure contour under gravity loading, the pressure contour scale is illustrated in the top left corner with units N/mm^2.

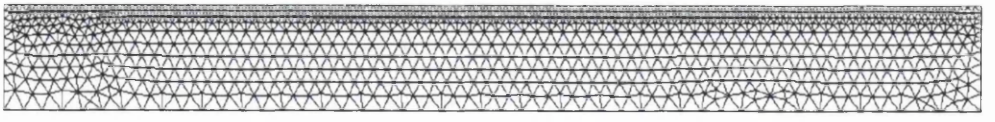


Figure 5.3a. Initial mesh

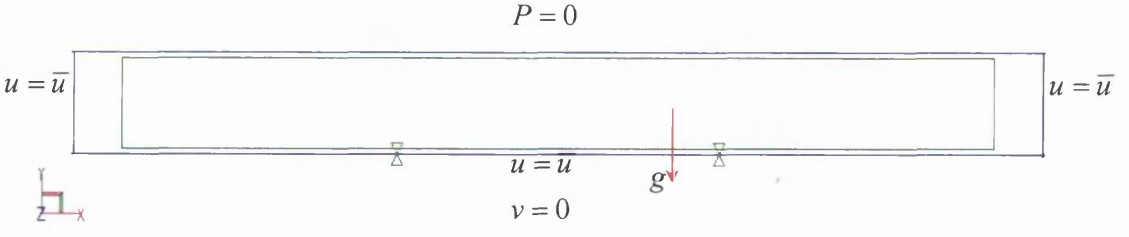


Figure 5.3b. Boundary conditions

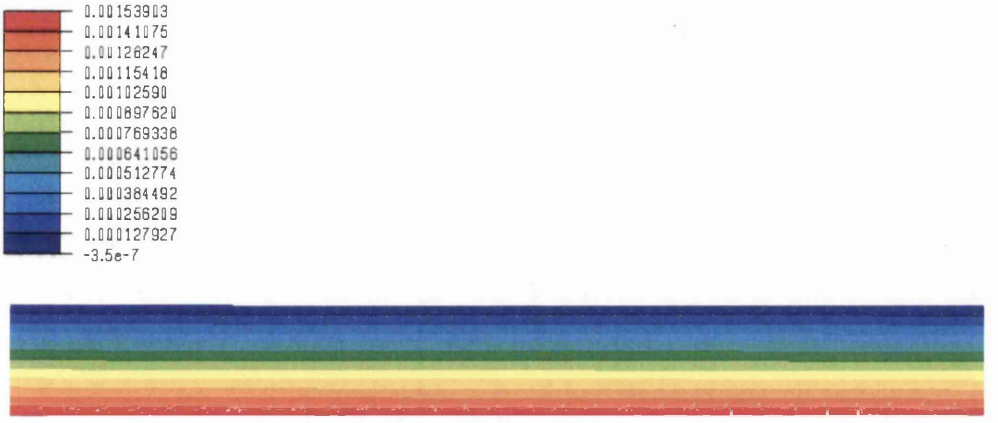
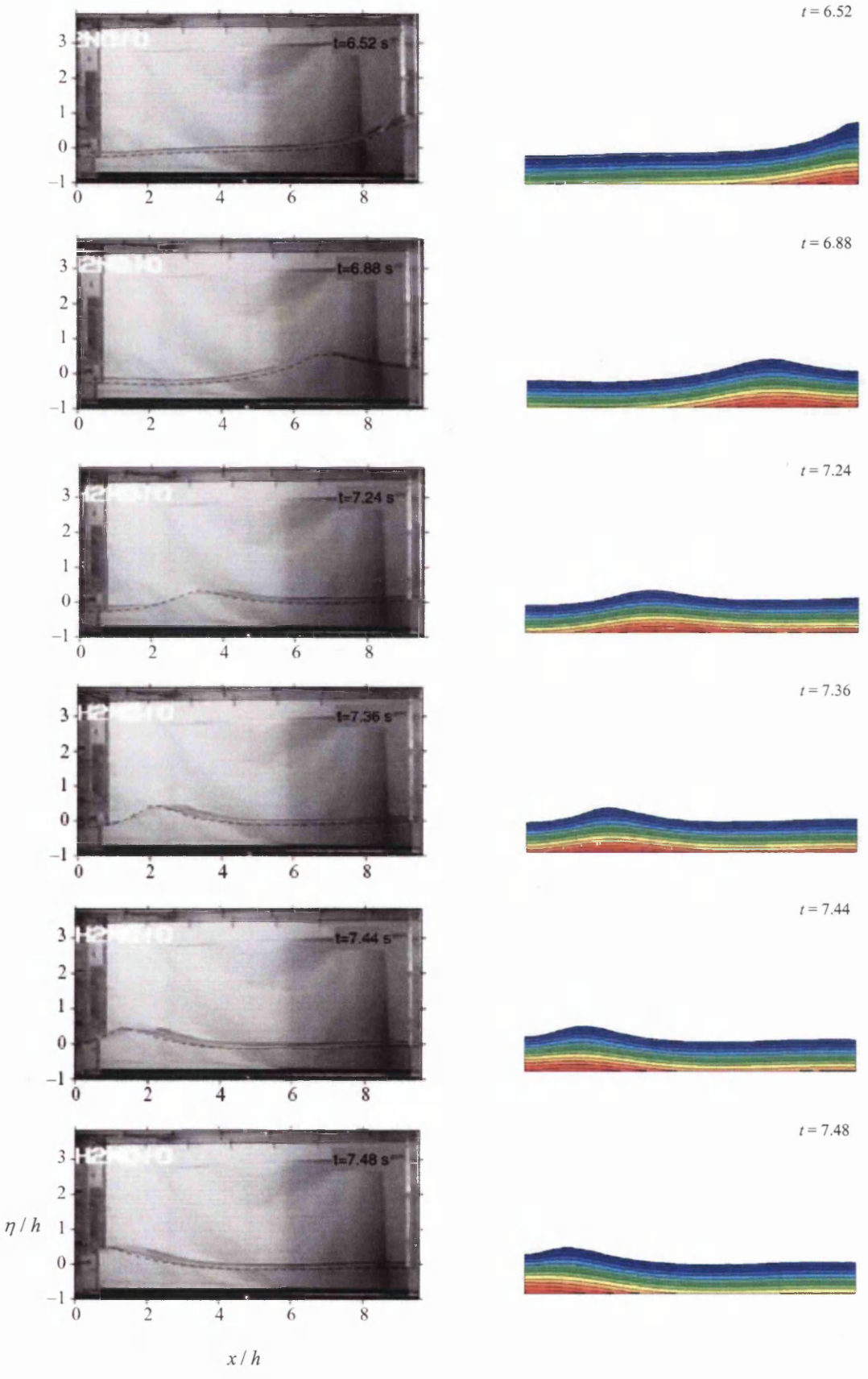


Figure 5.3c. Initial fluid pressure

The numerical results are compared directly with a selection of photographic snapshots at equivalent times during the tank excitation, at t = 6.52, 6.88, 7.24, 7.36, 7.44, 7.48, 7.52, 7.64 and 7.76 seconds. The location and amplitude of the principal wave at these instants encompass the full range of the wave excitation as it builds to the end of the loading period. The wave propagation profiles and pressure contours at successive times from the numerical simulation are compared against the experimental camera snapshots in Figure 5.4, where the vertical coordinate is ratio of standing wave height to initial height η / h, $\eta = z - h$, and the longitudinal coordinate is ratio of the distance to initial height x / h. It is noted that the dashed lines on the camera snapshots were from a numerical simulation conducted in reference [5.2]. It is seen that the agreement between the experimental and numerical results is very good. At time t = 6.88 and 7.24 seconds, the wave loses height while travelling across the tank. On the following images, the wave is seen to be steeper as it approaches the wall. A careful analysis comparison reveals that at time t = 7.64 seconds the experimental run-up on the left wall is about $\eta/h = 1.8$ compared with FE results of $\eta/h = 1.95$, and Bredmose's [5.2] results $\eta/h = 2.5$.


t=6.52 s
t=6.88 s
t=7.24 s
t=7.36 s
t=7.44 s
t=7.48 s
t = 6.52
t = 6.88
t = 7.24
t = 7.36
t = 7.44
t = 7.48
η / h
x / h

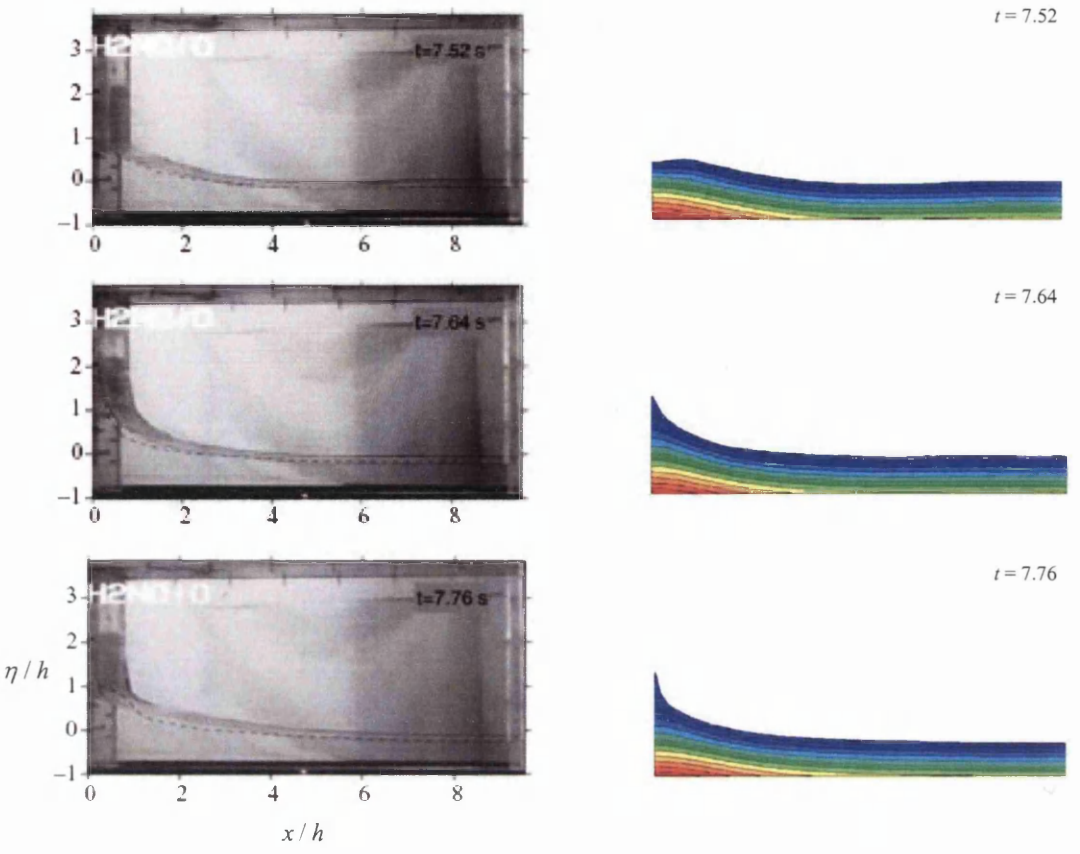


Figure 5.4 Comparison of experimental and numerical results. (On the left hand side the experimental results are shown with the numerical free-surface elevation plotted as a dashed line, which were conducted in reference [5.2].)

5.1.3 3-D implicit finite element modelling of horizontal water sloshing

The 2-D finite element modelling in section 5.1.2 can be extended to the 3-D case with a 200 mm width simulating the half width of the water tank. The initial finite element mesh consists of 25240 four-noded tetrahedral elements and 5673 nodes as shown in Figure 5.5a. The fluid geometry, initial boundary conditions, prescribed velocity and gravity loading are shown in Figure 5.5b. Only the fluid is simulated in the finite element model, the water tank itself is not modelled. The liquid is initially at rest and sloshing movement of the liquid is excited by the horizontal oscillation plus a push as in the same manner as the 2-D case. In the example, the properties of liquid are given in Table 5.1 and the stabilisation constant $\alpha = 1.0 \times 10^3$. A constant time step size of $\Delta t = 0.01$ is applied with error estimate check in every 5 steps intervals. The

element Jacobian based distortion error indicator is used to trigger the adaptive remeshing. An allowable Jacobian distortion error of 10% is specified in the test. The weighted least-square mapping scheme is introduced to transfer the nodal velocity and pressure to a new mesh and the closest neighbouring points N is chosen as 20.

The wave profiles and pressure contours at successive times from the numerical simulation are compared with the experimental camera snapshots in Figure 5.6, at time instants $t =$ 6.52, 6.88, 7.24, 7.48, and 7.88 seconds. Generally, the agreement between the experimental and numerical results is very good. The 3-D simulation has accurately predicted the motion of the principal wave, in amplitude and period.

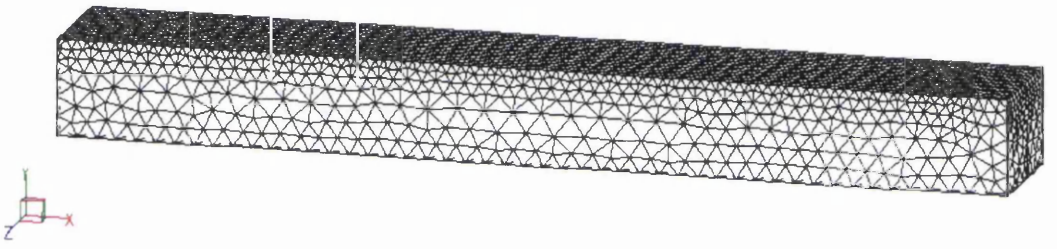


Figure 5.5a A 3-D initial mesh

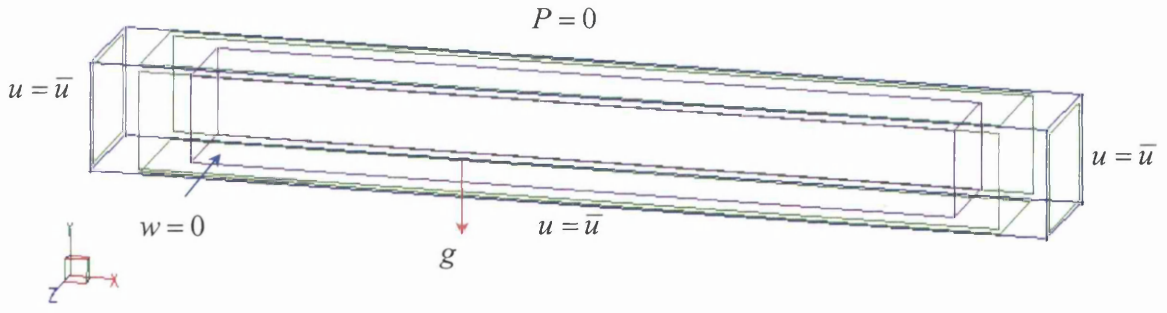


Figure 5.5b. Boundary conditions and loading

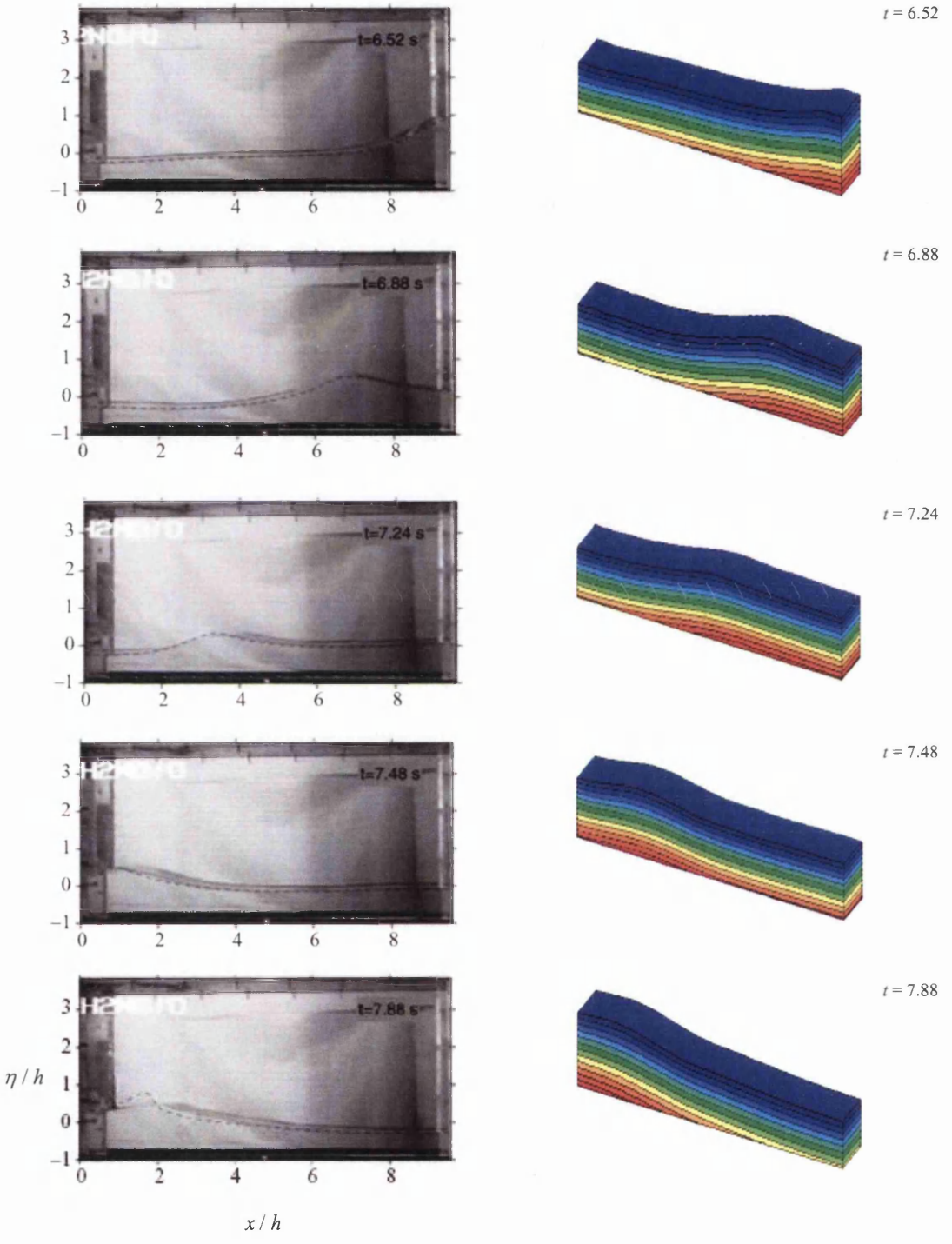


Figure 5.6 Comparison of experimental and numerical results
(Right hand side shows the fluid pressure at various time intervals)

5.1.4 2-D explicit finite element modelling of horizontal water sloshing

In the 2-D explicit finite element modeling of the water sloshing in a tank, the model is also specified as a plane strain problem and the tank is assumed to be a rigid solid. The fluid geometry is discretized into 1423 triangular linear elements, as shown in Figure 5.7. Smaller elements are generated at the top free surface in order to accurately capture the distortion of the wave generated. The mesh of the tank remains unchanged throughout the analysis. The prescribed velocity is applied on the rigid tank wall and gravity force is only applied to the fluid. The discrete elements contact is applied between the fluid and the rigid tank wall using a 2-D node-to-facet contact algorithm. A frictionless contact force is applied in the tangential direction of the facet and no penetration of the nodes is allowed in the normal direction. The critical time step Δt_{crit} is automatically adjusted according to the wave speed and a minimum length of the deformed fluid element. Its value is about $0.29 \sim 0.6 \times 10^{-6}$ seconds throughout the whole analysis and the time step reduction factor $\eta = 0.7$. The total number of time steps are around 11.5 million steps over 7.6 seconds of analysis. The fluid properties are kept the same and the stabilization constant $\alpha = 0.5 \times 10^4$. The error estimate is checked at every 4000 times steps with an allowable distortion angle of 5 degrees.

To validate the 2-D explicit modeling of the wave sloshing within the tank, the numerical results are compared directly with photographic snapshots at time instants $t = 6.52$, 6.88, 7.24, 7.48, 7.52, 7.64 seconds as shown in Figure 5.8. At time t = 7.64 seconds, the ratio of the standing wave height to initial height on the left side wall $\eta/h = 2.5$, compared with experimental results $\eta/h = 1.8$. Since the explicit analysis uses a very small time step and induces higher frequency responses, these are normally filtered in the implicit dynamic analysis.

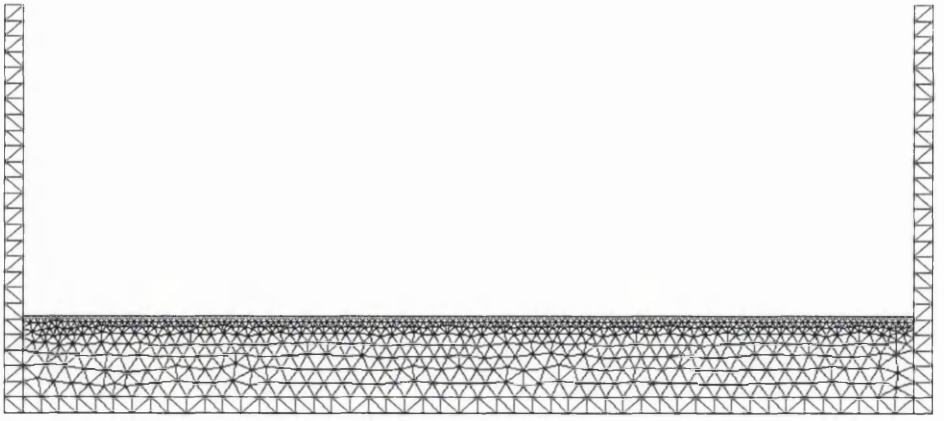
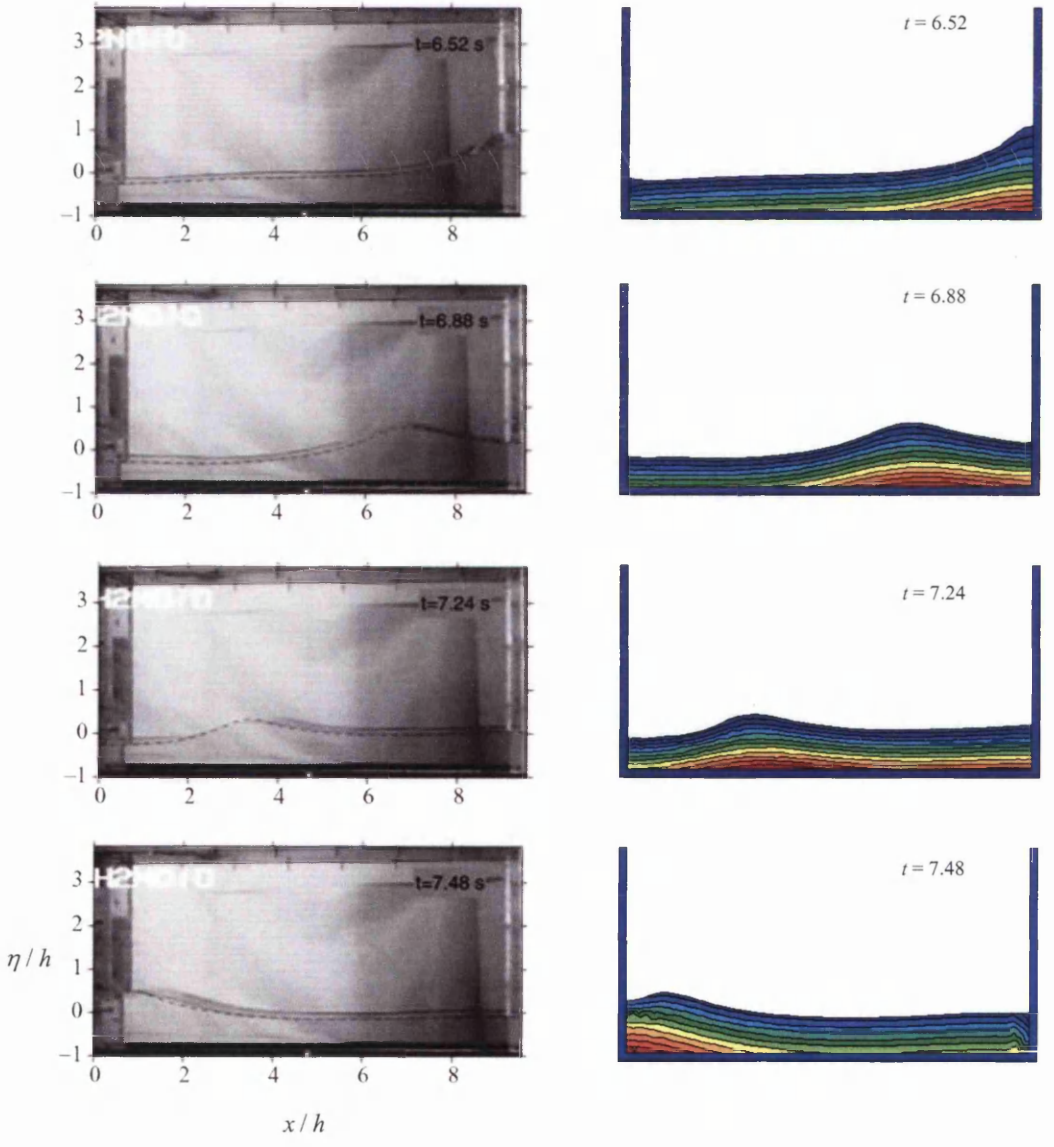


Figure 5.7. Initial mesh



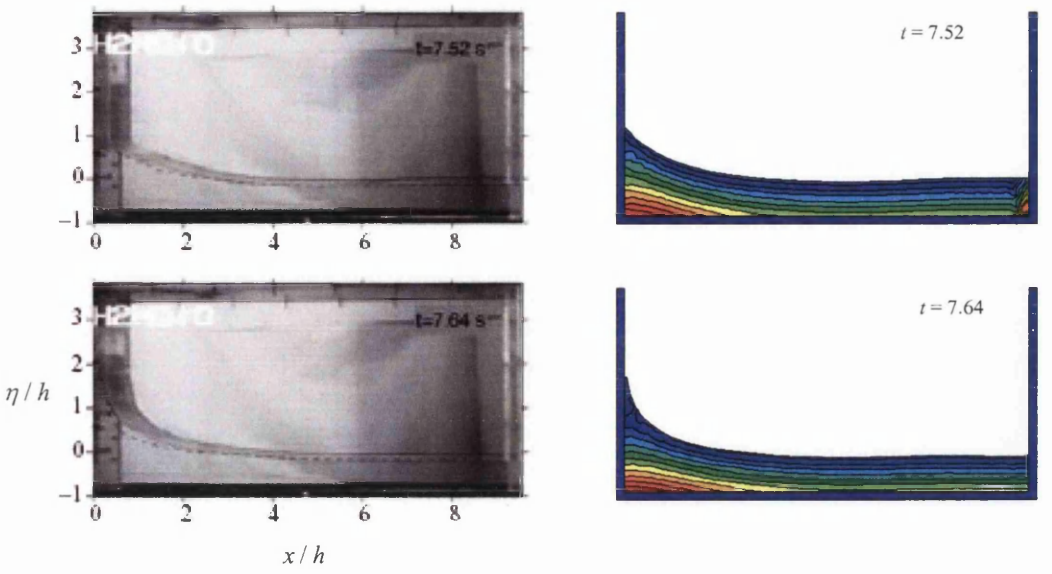


Figure 5.8 Comparison of experimental and numerical results
(Right hand side shows the fluid pressure contours at various time instants)

5.1.5 2-D implicit finite element modelling of vertical water shaking

In this section the numerical modelling for a vertically forced experiment test case is presented. The numerical model corresponds to the experiment case labelled V21 [5.2], for which the water depth was set H = 302 mm. A driving signal for experiment V21 consists of a small horizontal shaking of the tank, followed by a vertical oscillating motion. The acceleration signal used for modelling is based on the analytical expression adjusted to match the measured acceleration; it is expressed as

Horizontal acceleration

$$\ddot{\bar{u}}_x = \frac{10\pi^2}{(0.406)^2}\sin\left(\frac{\pi t}{0.406}\right) \qquad 0 \le t \le 2.4 \qquad 5.1a$$

Vertical acceleration

$$\ddot{\bar{u}}_y = -\frac{10\pi^2}{(0.203)^2}\sin\left(\frac{\pi(t-2.44)}{0.203}\right) \qquad 2.44 \le t \le 10.0 \qquad 5.1b$$

Figure 5.9a shows that a small peak followed by a decaying acceleration is seen in the horizontal signal around t = 2.3 seconds, it is due to the sudden stop of the horizontal

motion before the vertical motion begins. After the horizontal motion is stopped, the vertical movement of the tank is prescribed as in Figure 5.9b.

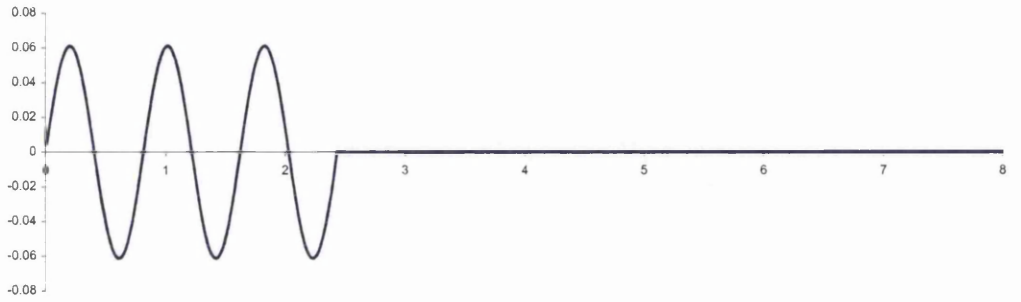


Figure 5.9a Horizontal acceleration/ g_0 against time

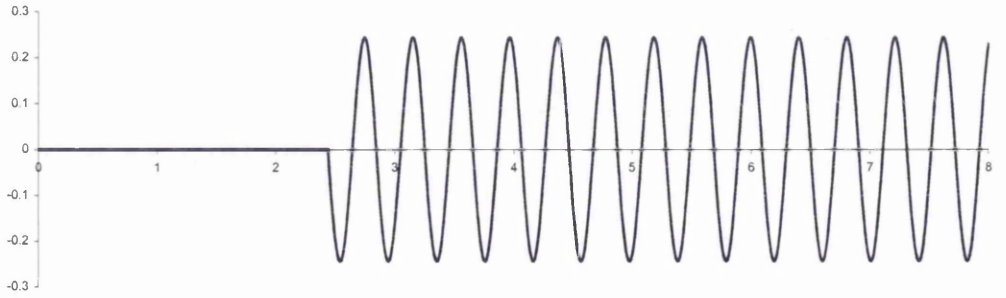
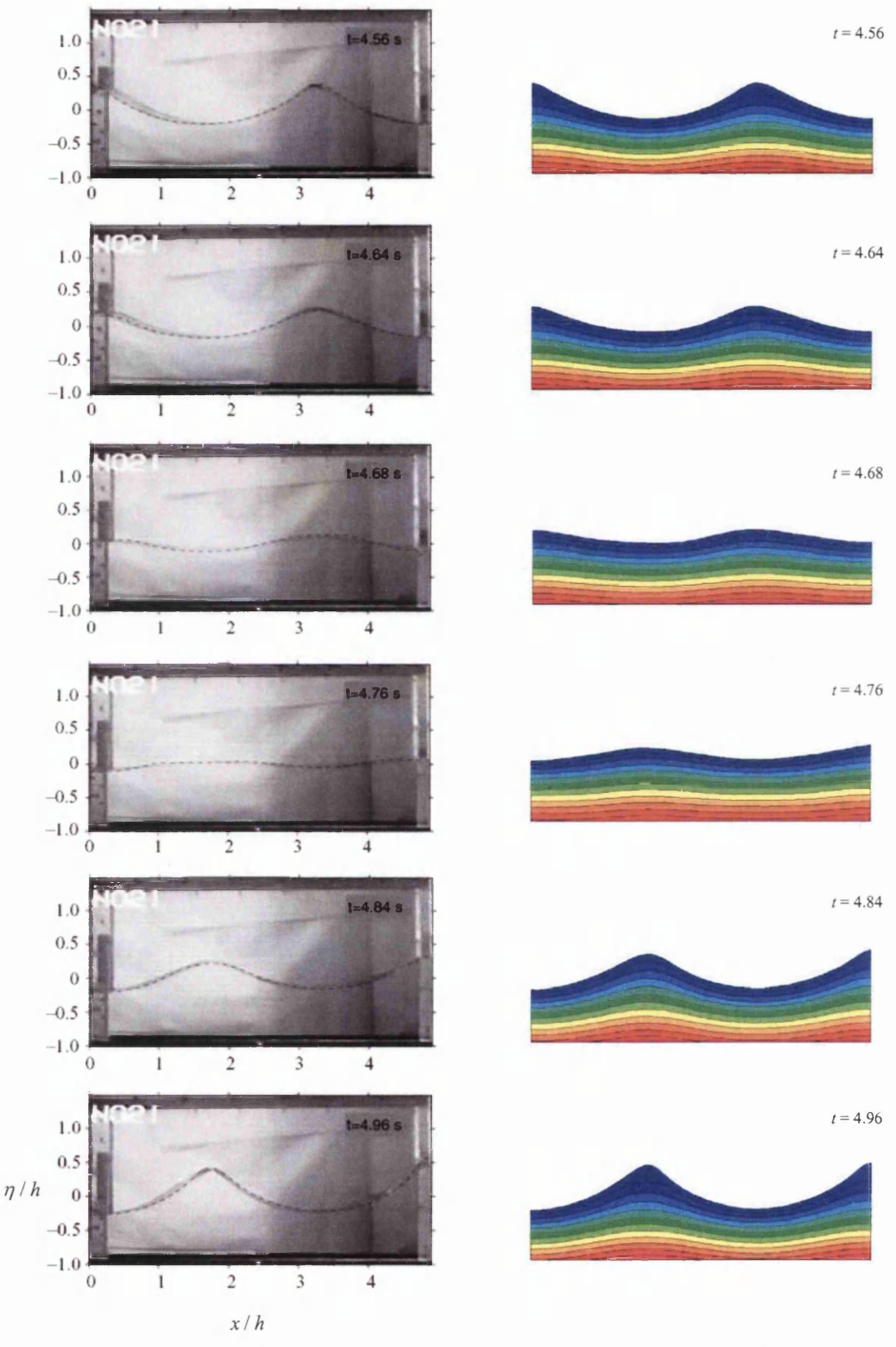
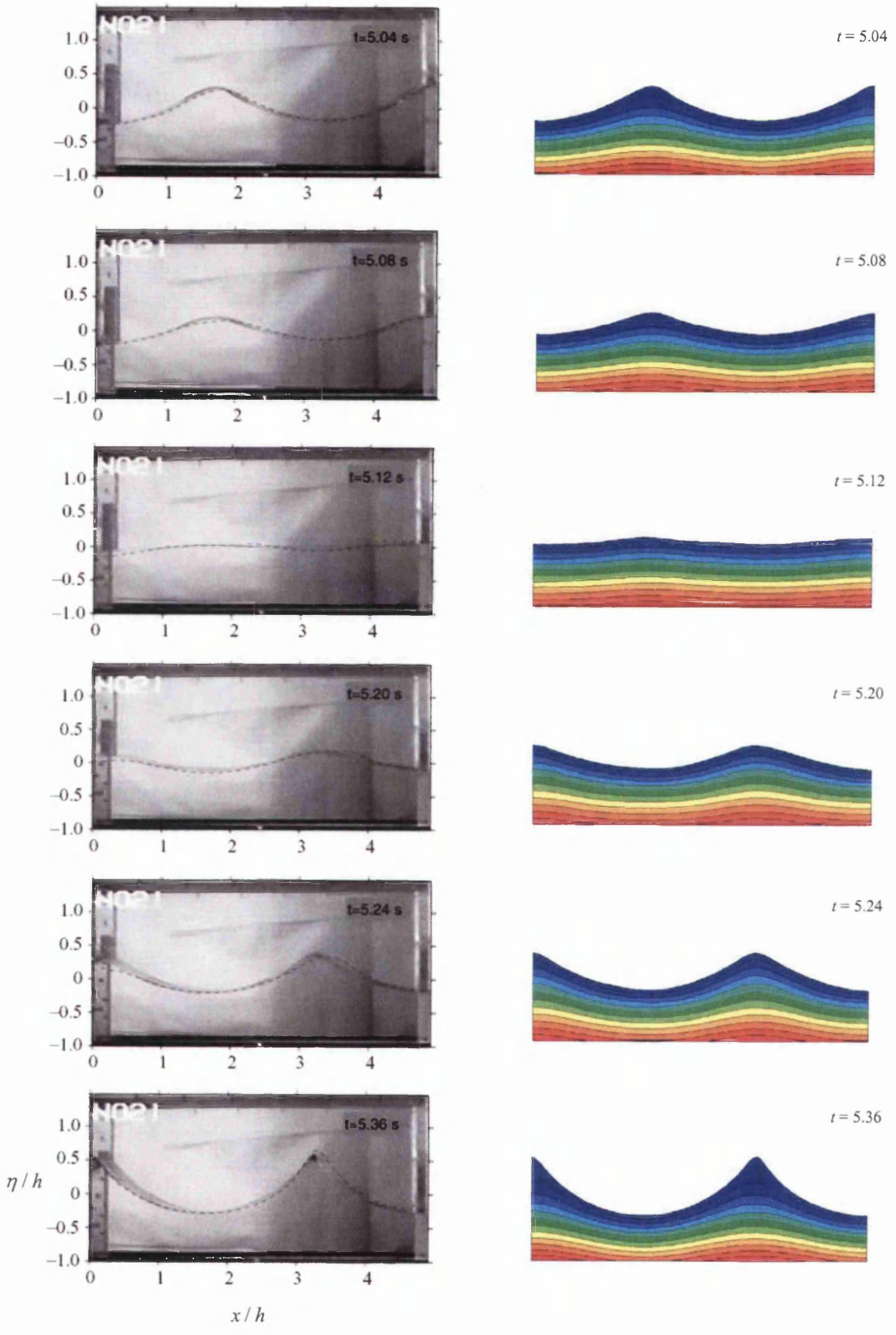


Figure 5.9b Vertical acceleration/ g_0 against time

When simulating experiment V21, an initial finite element mesh of 1965 linear triangle elements and 1094 nodes is employed and the material properties of the water is given in Table 5.1. The whole analysis is divided into two stages; at the first stage, a prescribed velocity curve $\bar{u}_x(t)$ is applied on the lateral side of the domain until t = 2.4 seconds, at the second stage the vertical prescribed velocity curve $\bar{u}_y(t)$ is applied on the base of the mesh. A constant time step of 0.01 second is used and the error estimate is checked at every 10 steps.


t=4.56 s
t=4.64 s
t=4.68 s
t=4.76 s
t=4.84 s
t=4.96 s
1.0
0.5
0
−0.5
−1.0
0
1
2
3
4
η / h
x / h
t = 4.56
t = 4.64
t = 4.68
t = 4.76
t = 4.84
t = 4.96


t=5.04 s
t=5.08 s
t=5.12 s
t=5.20 s
t=5.24 s
t=5.36 s
1.0
0.5
0
−0.5
−1.0
0
1
2
3
4
η / h
x / h
t = 5.04
t = 5.08
t = 5.12
t = 5.20
t = 5.24
t = 5.36

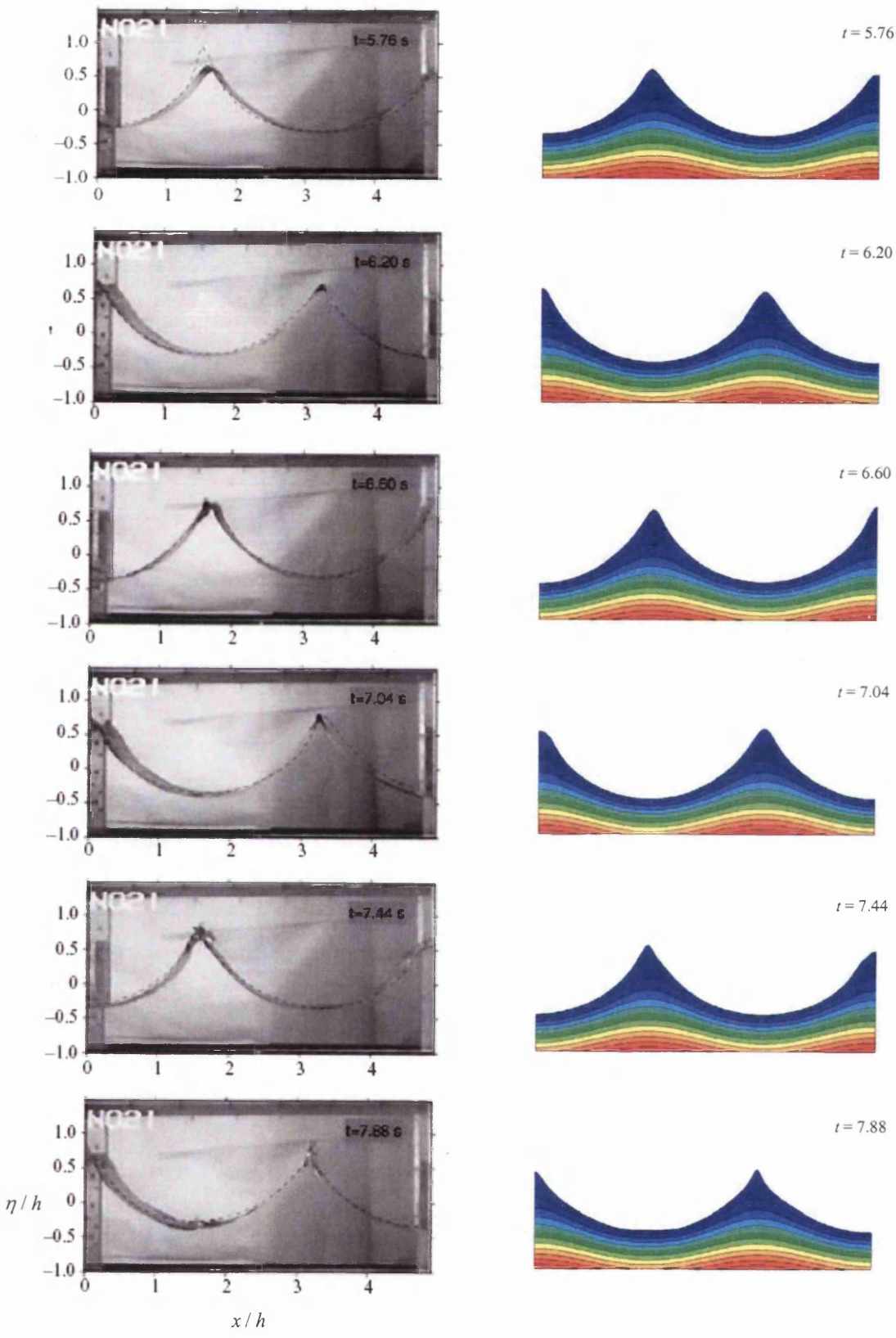


Figure 5.10 Comparison of experimental and numerical results
(Right hand side shows the fluid pressure at various time intervals)

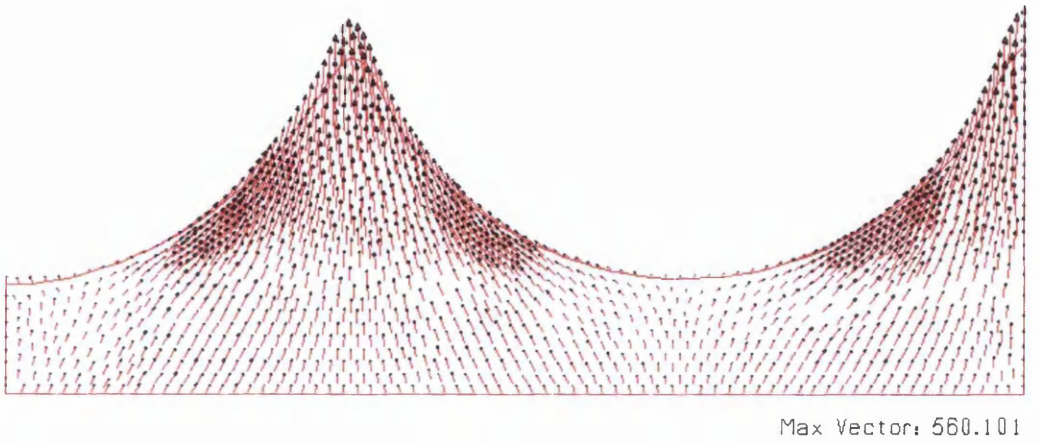


Figure 5.11 Velocity vector plot $u_{xy} = \|\boldsymbol{u}\|$ (mm/s), at time $t = 6.60$s

In Figure 5.10, eighteen images of the wave propagation are presented with the corresponding finite element solution plotted on the right hand side. The time values are given in the upper right corner of each image. The numerical deformed profiles with pressure contours compare very well with the experimental snapshots. The wave motions are much steeper and more violent than that under horizontal oscillation only. At time $t = 5.76$ seconds the height of the standing wave of the free surface is slightly over-predicted by both the FE analysis and the Boussinesq model, which is plotted as a dashed line on the image pictures. At time $t = 6.60$ second the maximum crest elevation is reached, and Figure 5.11 shows a velocity vector plot at this stage. After that, the downward motion of the following crest is depicted in the last three frames, and the wave heights are predicted well. The finite element analysis is stopped at time $t = 8.0$ seconds. The total CPU time of the whole simulation was 18 minutes and 12 seconds.

5.1.6 Simulation of liquid sloshing within a fragrance bottle

The finite element modelling of a liquid sloshing within a 'Roma' perfume bottle, which is being transported along a filling line, is analysed based on the explicit Lagrangian fluid formulation; Figure 5.12 displays the external profile of a 'Roma' perfume bottle.

Figure 5.12 External container profile of 'Roma' perfume bottle

On the filling line, the container wall is subjected to a horizontal velocity loading, corresponding to the velocity-time loading curve given in Figure 5.13. The fluid within the bottle is also subjected to a gravity force. Figure 5.14 illustrates an initial numerical geometry of the perfume liquid in the container, which is produced by using CAD drawing due to its complicated geometrical shape. The properties of the perfume liquid are identical to water, and are given in Table 5.1. The discrete element contact algorithm defined in chapter 3 is adopted to simulate a frictionless contact interaction between the liquid and the container wall. The aim of modelling is to assess the possibility of liquid splashing out of the container on a filling line.

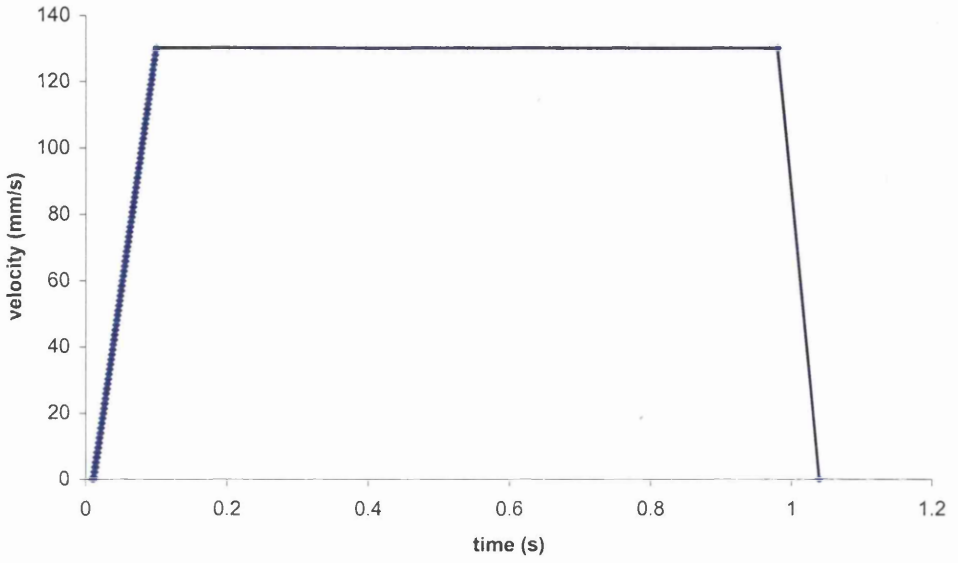


Figure 5.13 Applied velocity versus time

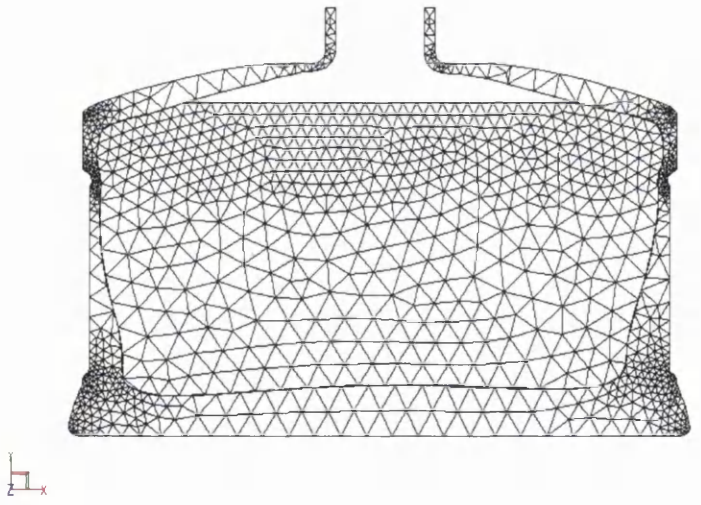


Figure 5.14 Initial numerical geometry

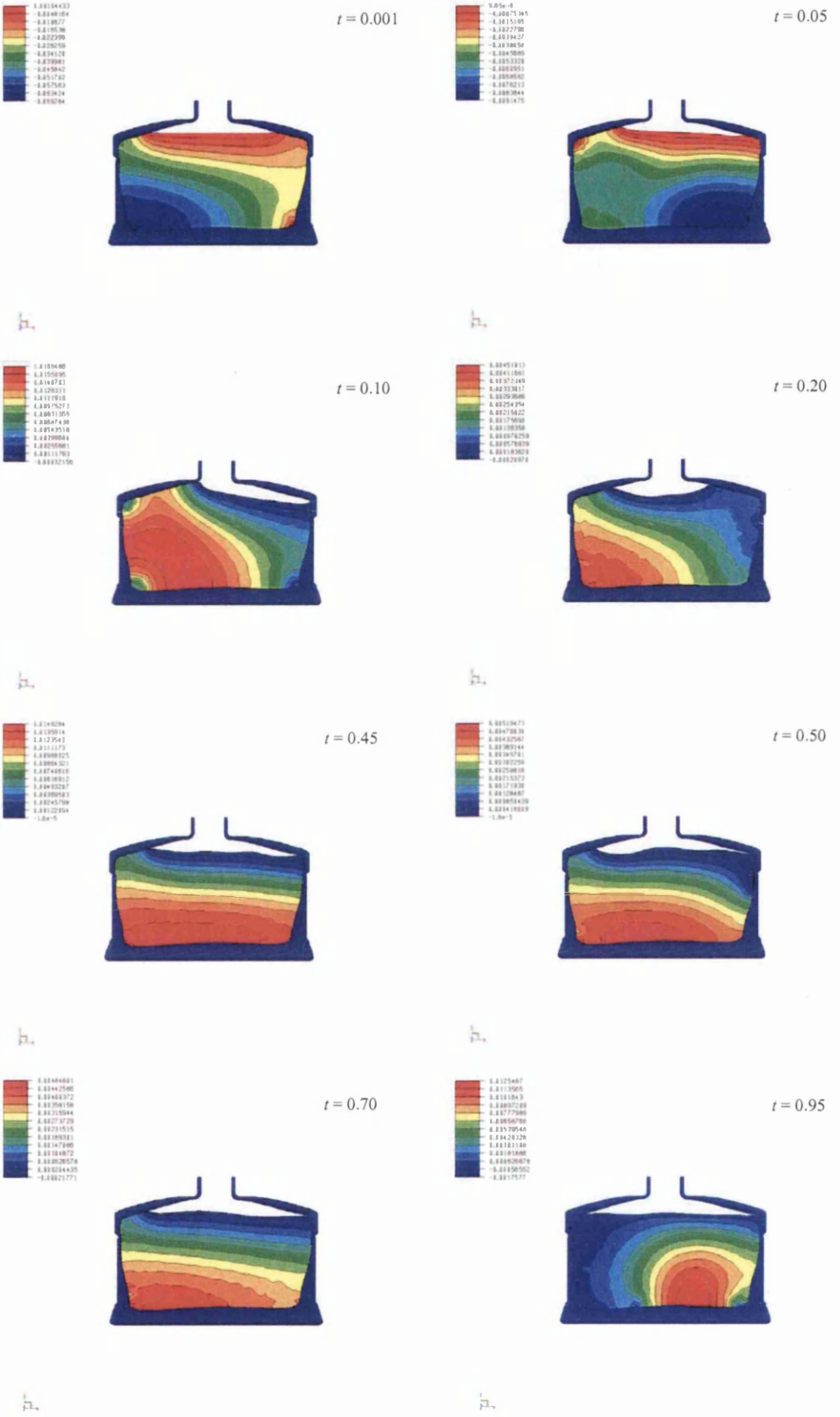


Figure 5.15 Fluid pressure at various time instants

The critical time step Δt_{crit} is defined by the wave speed of the liquid and a minimum length of the deformed fluid element; its value is about 0.2×10^{-6} seconds throughout the analysis. The perfume bottle wall is defined as solid elements, the properties of the solid are adjusted, so that the critical time step is not governed by the solid elements. The time step reduction factor η is 0.7. The total number of time steps is around 4.22 million steps for over 1.0 seconds of analysis, with a total CPU time of 13 hours and 19 minutes. The stabilization constant is $\alpha = 0.1\times10^{3}$. The error estimate is checked at every 2000 times steps with an allowable distortion angle of 5 degrees.

The wave profiles with pressure contours at successive times from numerical simulation are presented in Figure 5.15. The maximum standing wave, which reaches the container neck, occurs at time 0.1 seconds, corresponding to the highest velocity $\overline{u}_x$=130 mm/sec at that time. At the constant velocity stage the standing wave begins to gradually recede.

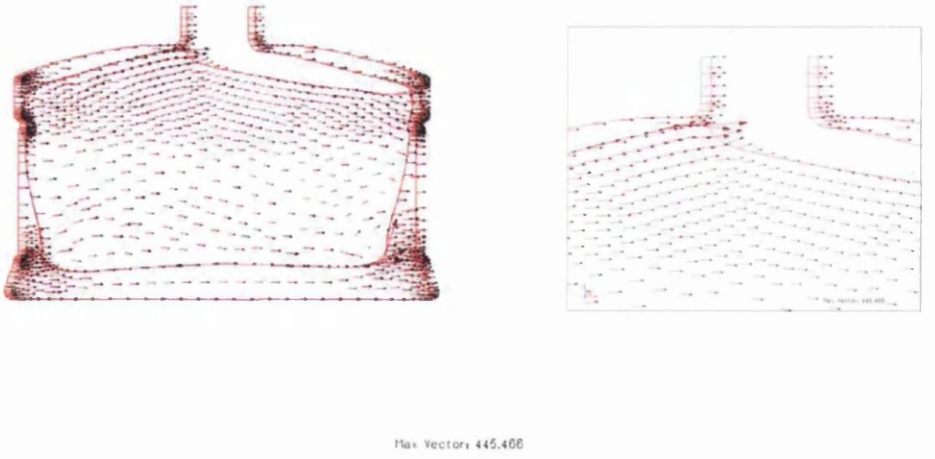


Figure 5.16 Velocity vector plot $\overline{u}_{xy} = \|\boldsymbol{u}\|$ (mm/s), at time t = 0.10s

Figure 5.16 illustrates the velocity vectors at 0.10 seconds, indicating a high probability of fluid splashing out of the container neck. This splashing is a result of the acceleration of the container from the stationary state and subjected to a maximum velocity within 0.1 seconds.

5.2 Collapse of a liquid column

The analyses of collapse of a viscous liquid column have been reported by several authors[5.4][5.5]. In this section, two numerical examples of collapse of a liquid column are considered; collapse of a 2-D axisymmetric liquid column and collapse of a 3-D liquid column, are presented to provide an assessment of the performance of the schemes proposed in chapter 2.

5.2.1 Collapse of a 2-D axisymmetric liquid column

An axisymmetric 2-D liquid column with dimension R350× H700 mm is under gravity loading, with the gravitation $g = 9800\,mm/\sec^2$. The boundary conditions are defined in Figure 5.17b, with $\bar{u}_x = 0$ at the axisymmetric line and $\bar{u}_y = 0$ at the base line of the liquid column. Zero pressure is set at both top and lateral surfaces. The liquid domain is initially divided into 888 axisymmetric triangular elements with a total number of 487 nodes, as shown in Figure 5.17a. The liquid material properties are given in Table 5.1. The stabilisation constant is set as $\alpha = 0.2\times10^2$. The time step is chosen as $\Delta t = 0.01$ second and the plotting files are output at 0.1 second time intervals. The total simulation time is 0.5 second.

The element distortion error check is set in every 5 steps with an allowable distortion angle of 5 degrees and the maximum and minimum allowable angles are 165 and 15 degrees, respectively. The error estimator is based on the energy norm, which is given by equation (3.64). The maximum and minimum mesh density of prediction is set as $l_{\max} = 35mm$ and $l_{\min} = 20mm$.

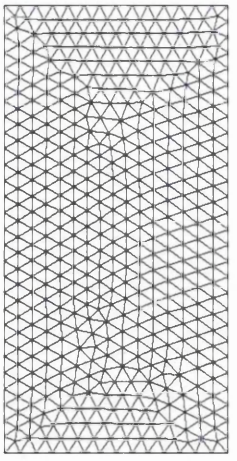


Figure 5.17a Initial mesh

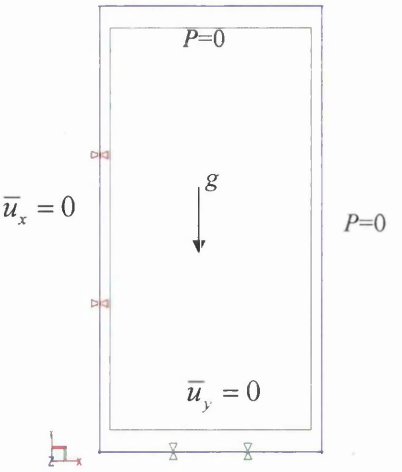


Figure 5.17b Boundary Conditions

Figure 5.18 shows the deformed profiles and pressure contours at time instants t = 0.1, 0.2, 0.3, 0.4, 0.5, seconds respectively. The ratio of height to base radius (H/R) at each time instant is also presented. From the results, it is evident that the stabilization constant α in the space-time Galerkin / least-squares formulation is not a sensitive value. The pressure results are almost the same when α is taken between $0.2\times10^2 \sim 0.2\times10^4$. However, it is seen in Figure 5.19 that when $\alpha = 0.2\times10^2$, the volume conservation is better, with only 1.9% volume change.

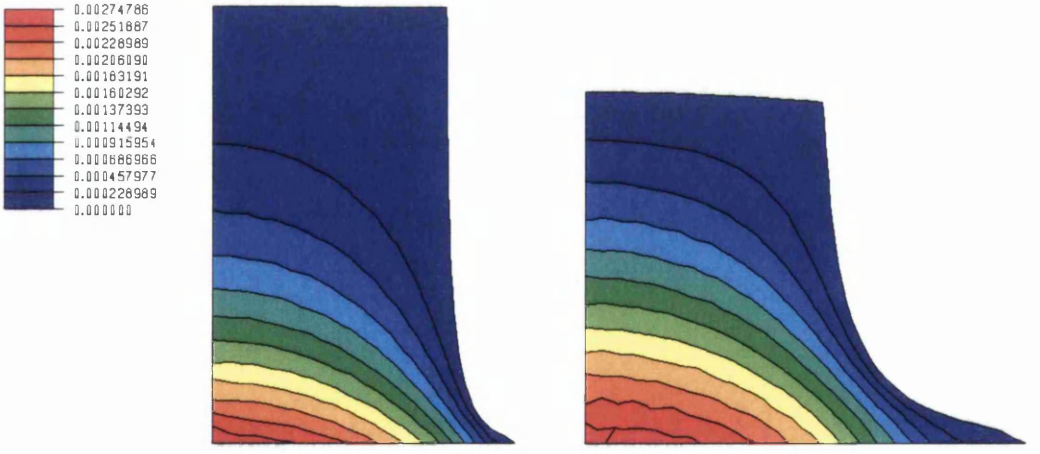


Figure 5.18a Pressure contour at $t = 0.1$ $(H/R) = 1.4715$

Figure 5.18b Pressure contour at $t = 0.2$ $(H/R) = 0.8081$

Figure 5.18c Pressure contour at $t = 0.3$, $(H/R) = 0.4060$

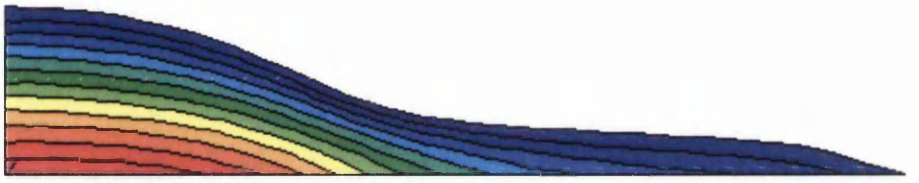


Figure 5.18d Pressure contour at $t = 0.4$, $(H/R) = 0.1884$

Figure 5.18e Pressure contour at $t = 0.5$, $(H/R) = 0.0858$

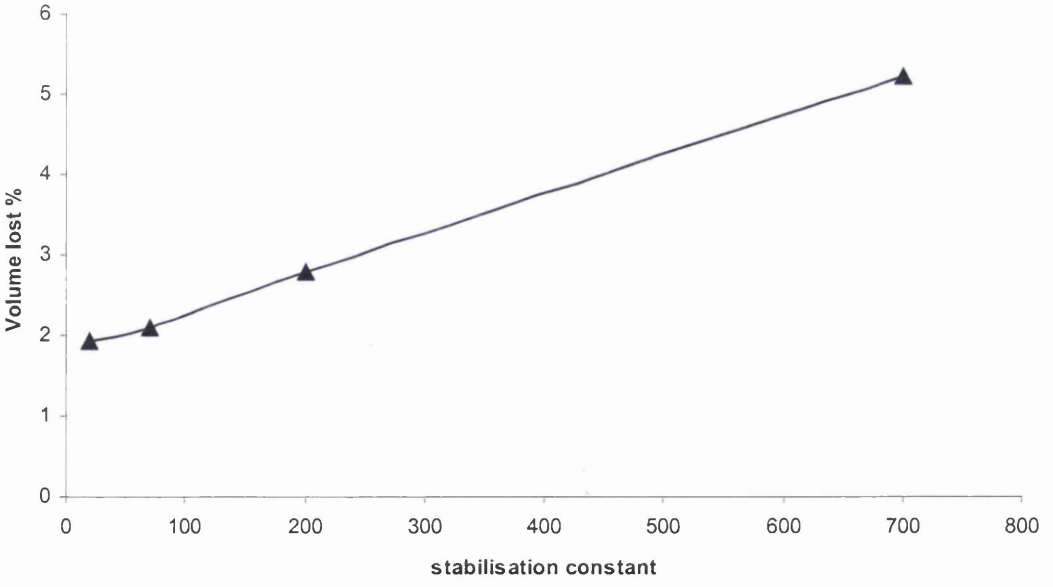


Figure 5.19 Rate of volume change versus α

5.2.2 Collapse of a 3-D liquid column

The collapse of a 2-D axisymmetric liquid column is extended to a 3-D case. Only a quarter of the cylinder is simulated with R350×H700 mm. Symmetric boundary conditions are set in the *x-y* and *y-z* plane of the cylinder. A liquid initial mesh consists of 23273 four-noded tetrahedral elements and 4860 nodes, as shown in Figure 5.20a. The liquid column is initially at rest and gradually spreads out under gravitational force. The 3-D problem is solved using the preconditioned Bi-CGSTAB iterative method, since a large system of equations is involved. In this example, the stabilization constant $\alpha = 0.2 \times 10^3$ is used.

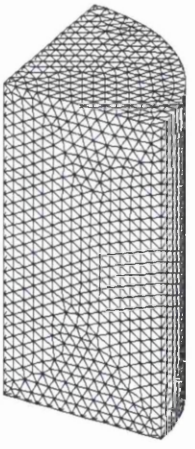


Figure 5.20a Initial mesh

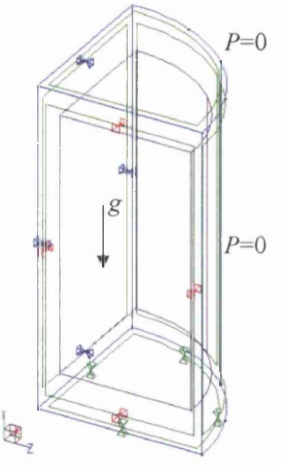


Figure 5.20b Boundary Conditions

The 3-D deformed profile and pressure contour at time instant t = 0.1, 0.2, 0.3, 0.4, 0.5 seconds are presented in Figure 5.21, with the 2-D deformed profile displayed at the right side of images for comparison. The ratio of height to base radius of the column is also given, it can be seen that both 2-D axisymmetric and 3-D simulations give identical results.

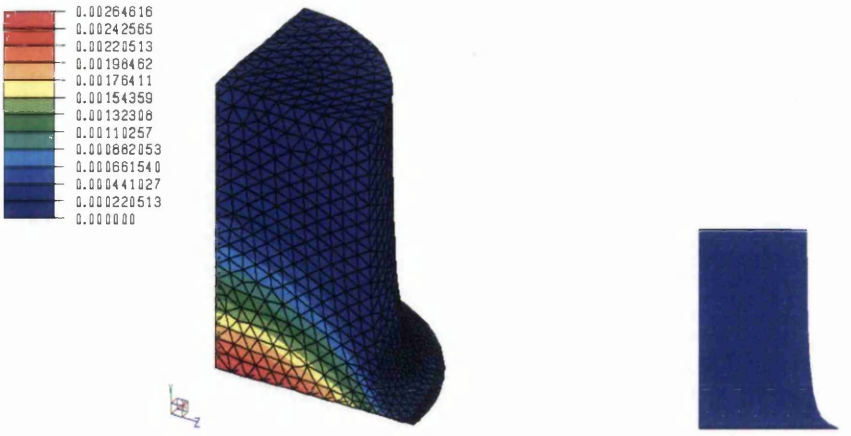


Figure 5.21a Pressure contour at $t = 0.1$, $(H/R) = 1.4739$

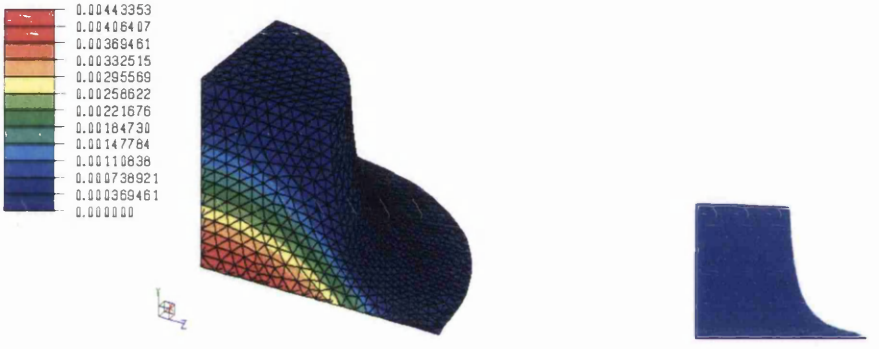


Figure 5.21b Pressure contour at $t = 0.2$, $(H/R) = 0.8128$

Figure 5.21c Pressure contour at $t = 0.3$, $(H/R) = 0.4042$

Figure 5.21d Pressure contour at $t = 0.4$, $(H/R) = 0.1955$

Figure 5.21e Pressure contour at $t = 0.5$, $(H/R) = 0.08994$

5.3 2-D explicit modeling of shampoo container filling

The objective of numerical simulation of a Dove shampoo container filling is to help the industry to understand possible liquid movement, whether air entrapment will occur or cause spillage of the liquid during the filling process. Finite element modelling is based on the explicit fluid dynamic formulation, since simulation involves complicated contact interactions among fluid particles and between fluid particles and the rigid container walls.

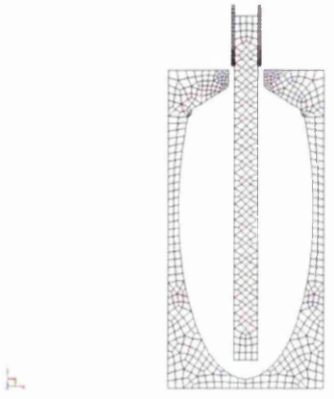


Figure 5.22a Initial mesh

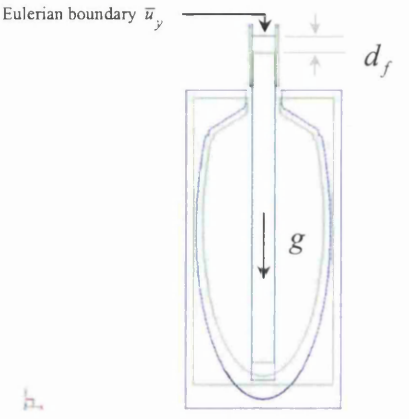


Figure 5.22b Boundary Conditions

The geometry of a 200ml Dove container with outside size $H0.1613 \times W0.08$ m is shown in Figure 5.22. The liquid is filled into the container via a nozzle, which is about 7 mm in diameter and positioned above the finish of the container. The finite element analysis is set as a plane strain problem. A special Eulerian boundary is designed to simulate the liquid filling; as the fluid passes a predefined distance d_f from the top line of the nozzle, extra fluid is added within the nozzle, i.e. the top line of the nozzle is pulled back to its original position, and the fluid domain is re-meshed. A prescribed filling velocity $\bar{u}_y = -1.458m/\sec.$ is applied on the nodes of the top line of the nozzle. At the same time the fluid elements are subjected to gravity load with $g = 9.8m/\sec^2$. The viscosity of the fluid material is known to be shear rate dependent and the Herschel-Bulkley law is used to simulate a shearing thinning behaviour of the shampoo liquid. The properties of the Dove shampoo liquid are listed

in Table 5.2 and the experimental and numerical fit of the viscosity versus strain rate of the Dove DMM14 material is shown graphically in Figure 5.23.

Property	*Value*
Bulk modulus	$K = 1.075\times10^{10}$ Pa
Density	$\rho = 1.0\times10^{3}$ Kg/m^3
Yield Stress	$\sigma_y = 0.5$ $Pa\cdot\sec$
Fluid consistency index	$\mathbb{C} = 21.0$ $Pa\cdot\sec$
Power law index	$n = 0.25$
Critical shear rate	$\dot{r} = 0.002$ $1/\sec$

Table 5.2 Material properties

The discrete element contact of the 2-D node-to-facet contact algorithm is applied among the fluid particles and between the fluid particles and the rigid container walls. The frictional coefficient c is set as 0.2 for the Coulomb friction law. The normal and tangential penalty value is 1.0×10^{8} and 1.0×10^{7}, respectively. The critical time step Δt_{crit} is about $0.5\sim1.0\times10^{-6}$ seconds throughout the analysis, the time step reduction factor $\eta = 0.7$. The total number of time steps is around 0.6 million steps for over 0.38 seconds of analysis. The stabilization constant is set as $\alpha = 0.5\times10^{4}$. The error estimate is checked at every 500 times steps with an allowable distortion angle of 5 degrees. The maximum and minimum element size of a new adaptive mesh is defined as $l_{\max} = 0.0032$ and $l_{\min} = 0.002$ m.

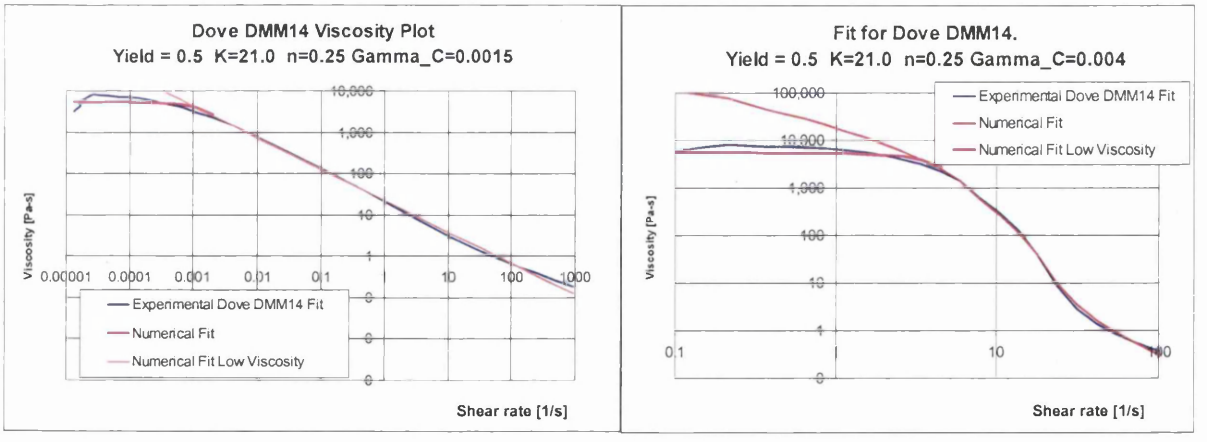


Figure 5.23 Dove DMM14 experimental and numerical material model fit

Figure 5.24 2-D Shampoo filling; velocity vector plot at various time intervals

Figure 5.24 shows a continual shampoo filling process and eight liquid profiles with velocity vector plots are given at successive times. The initial lateral side of the filling liquid column is parallel. As the filling process continues the lower part of liquid column becomes thinner due to gravity force. This is a real phenomenon observed at the filling site. As the shampoo fluid gradually fills up the container, shear thinning of the liquid column is reduced and internal folding of the liquid appears at time 0.325 seconds. At that time two trapped air pockets are formed. As we can see in Figure 5.25, self-contact among the fluid particles is well simulated by the discrete element contact algorithm. Obviously, within a real fluid domain these folds do not exist, and can be removed by stitching element surface boundaries and redefining the nodes on the surface geometry entity during the adaptive remeshing procedure. The 2-D numerical simulation gives a good indication of how the container filling can be achieved.

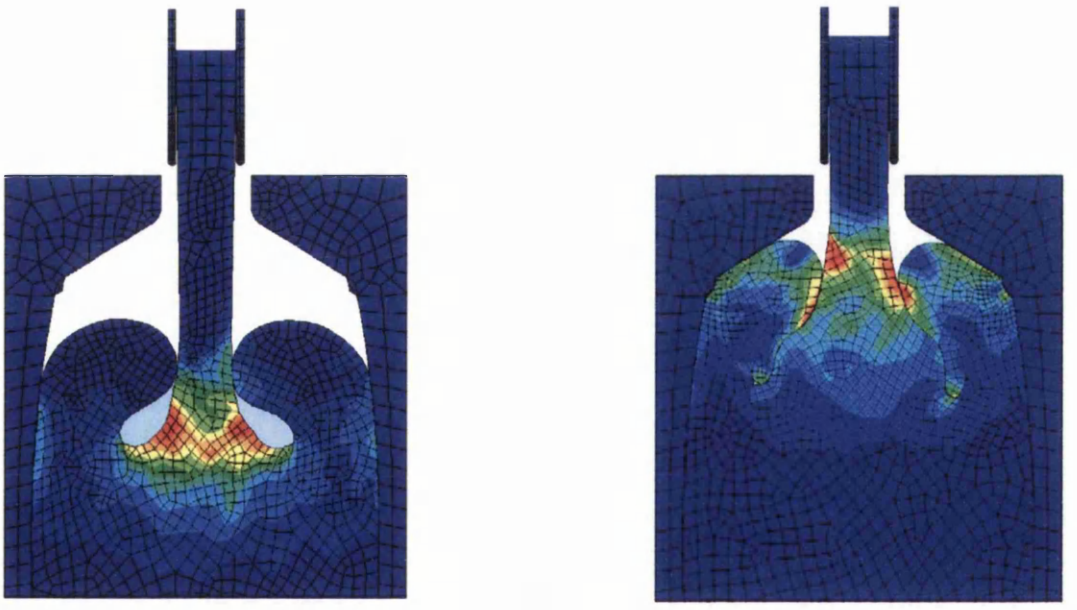


Figure 5.25 Effective strain rate at time $t = 0.325$s and $t = 0.38$s

5.4 Parallel performance on a distributed memory platform

To evaluate the parallel performance of the algorithm implemented in chapter 4, two of the most common measurements; parallel speed-up and efficiency are used and expressed as

$$S(N_p) = \frac{T(N_1)}{T(N_p)} \tag{5.2}$$

$$e(N_p) = \frac{S(N_p)}{N_p} \tag{5.3}$$

Where $T(N_1)$ and $T(N_p)$ denote the execution CPU time of the algorithm using a single processor N_1 and multiple processors N_p, respectively. Ideally, a parallel algorithm running on N_p processors may run N_p time faster than on a single processor. However this cannot be achieved in practice as various overheads can compromise the final performance. These are most commonly associated with the sequential tasks in a parallel algorithm, the unequal distribution of computational load among the processors and the communication cost between the processors, which is a main factor of reducing parallel speedup on a distributed memory computer system.

Robust and efficient transmission of a database record between the master and slave processors is of paramount importance to the success of implementation of a parallel program. *'Remember it is much faster to communicate 1 data block of 100 components rather than communicating 100 data blocks each of 1 component.'* To fulfil the efficiency requirement ELFEN packaging C library [5.6] is used, which is developed based on the MPI library [5.7][5.8] and using 21 basic MPI functions to obtain essential communication functionality. The packaging routines include three critical steps:

- The packing of database records; it requires defining or creating of a package name and adding specific data records to the package.

- The communication of the package, sending and receiving of a package.

- The unpacking of the transmitted data records on the required database.

To implement the above tasks six basic commands are introduced into the ELFEN packaging C library, they are

PCKNAM	- creates a package name
PCKDEF	- defines the package
PCKADD	- adds specific data to the package
PCSEND	- sends the defined package to identified processors
PCRECV	- receive the defined package from an identified processor
PCSYNC	- package synchronisation

The tests in this section have been run on a PC based interconnection network system, which consists of several Intel Pentium PC nodes (1.8 GHz), each with 1.04 GB of memory. Each computing node is locally interconnected by a high-performance switch communication hardware (DES 3225G), which has a speed of 100 Mb/second and less than 20 μs latency. The processing nodes have been reserved for 'unique' CPU usage, i.e., only the test program was running on each node.

5.4.1 Implicit analysis of a simply supported T beam

A simply supported steel T-beam of a length 4000 mm is subjected to a uniform pressure loading, $P = 0.2\,N/mm^2$, on the top surface. Its cross section geometry and supporting boundary conditions are illustrated in Figure 5.26. A finite element mesh consists of 8585 nodes and 6200 eight-noded hexahedral F-bar elements [5.9][5.10], with $2\times2\times2$ Gauss integration points. A Von Mises elasto-perfectly plastic model with initial yield stress $\sigma_0 = 100N/mm^2$ is used to simulate possible plastic yielding. The material properties of the steel are defined in Table 5.3. Since the stiffness matrix produced in the F-bar formulation is a non-linear, unsymmetric one, the Bi-CGSTAB iterative solver is used in solving the resulting interfacial equations in the master processor. The test case is run on 1~4 processors, separately. The performance of parallel speed-up and efficiency is given in Table 5.4. Figure 5.27a shows the

separated effective stress contour of the T-beam with three subdomains from 4 processors running (1 master and 3 slave processors), compared with the combined results from sequential analysis in Figure 5.27b. Maximum displacement d_y and effective plastic strain $\bar{\varepsilon}_{vp}$ is –3.345 mm and 0.00765, respectively.

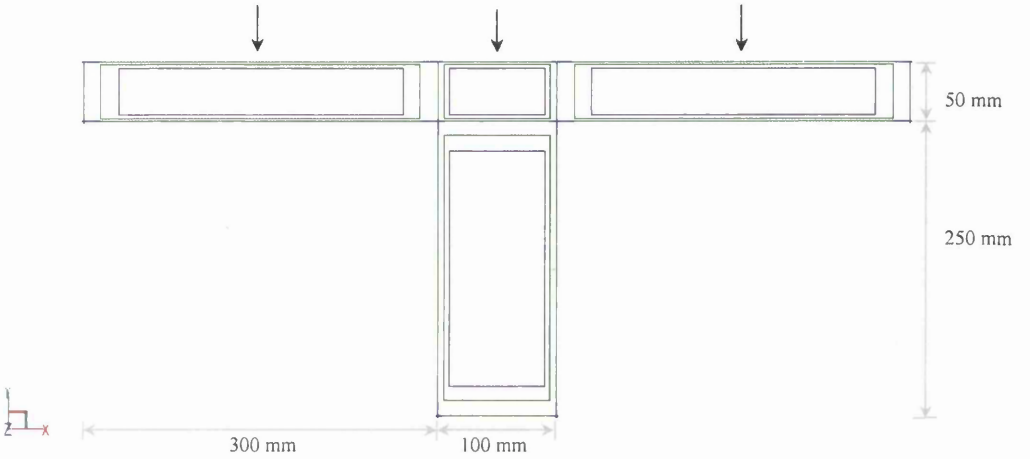


Figure 5.26a Cross section geometry of T-beam

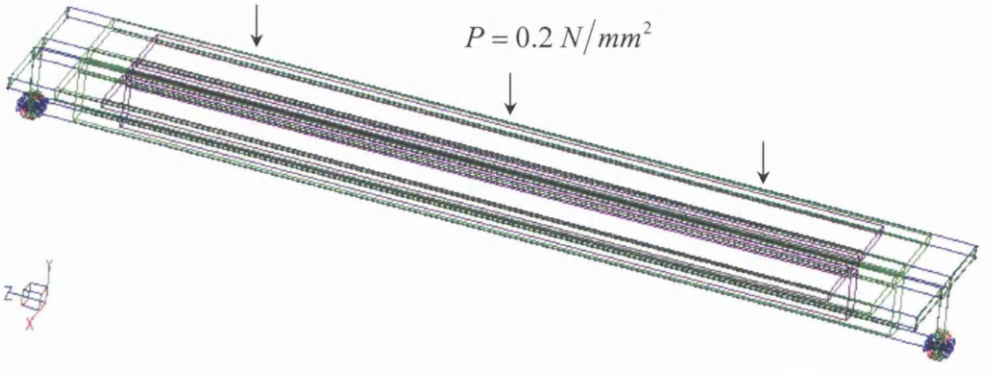


Figure 5.26b Boundary condition and pressure loading

Property	***Value***
Young's Modulus	$E = 2.1\times10^5\ N/mm^2$
Poisson's Ratio	$v = 0.29$
Density	$\rho = 7.86\times10^{-9}\ N\,\mathrm{sec.}^2/mm^4$
Initial yield stress	$\sigma_0 = 100N/mm^2$

Table 5.3 Properties of steel material

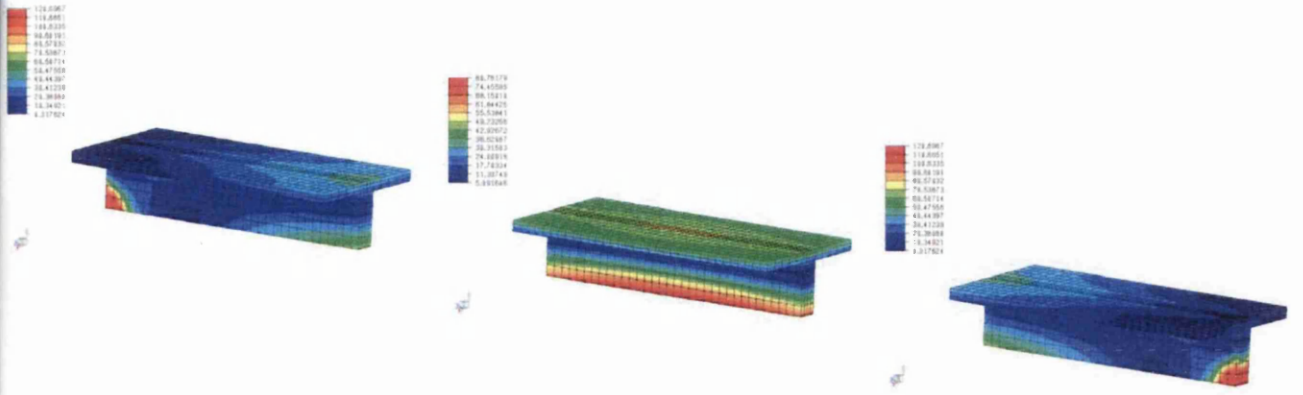


Figure 5.27a Effective stress contour of T-beam (3 subdomains)

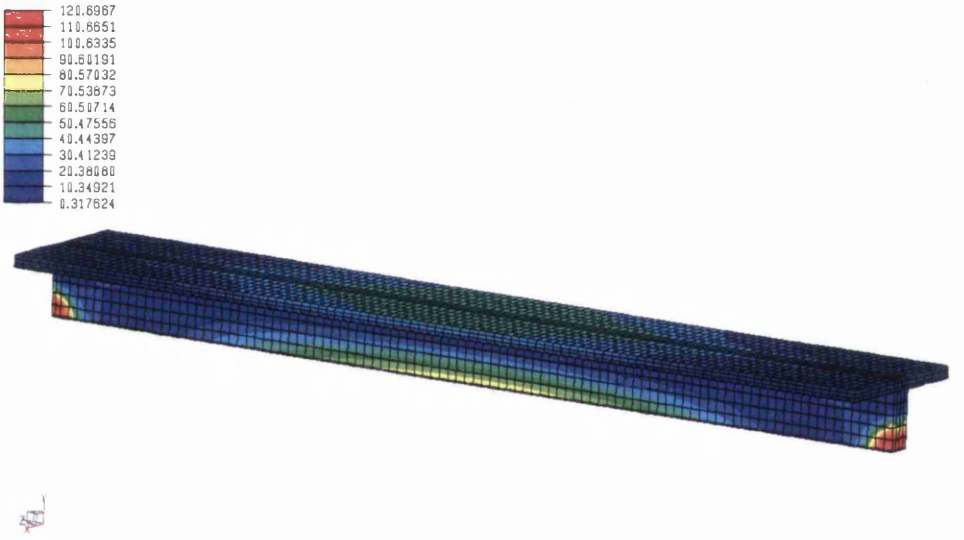


Figure 5.27b Effective stress contour of T-beam (combined)

Number of Processors	*Time* (s)	*Speed-up*	*Efficiency*
1	399.01	-	-
3	159.01	2.509	0.83644
4	116.12	3.436	0.85905

Table 5.4 Parallel speed-up and efficiency

Solver stages	*Iteration 1*	*Iteration 2*	*Iteration 3*	*Iteration 4*	*Iteration 5*	*Iteration 6*
Pre-solver	0.30	0.0	0.0	0.0	0.0	0.0
Waiting	11.06	11.25	11.20	11.22	11.22	11.22
Receive S_p	2.64	2.62	2.69	2.64	2.67	2.62
Assemble $S = \sum S_p$	0.25	0.25	0.25	0.25	0.22	0.27
Receive y_p	0.0	0.0	0.0	0.0	0.0	0.0
Assemble $y = \sum y_p$	0.0	0.0	0.0	0.0	0.0	0.0
Preconditioner	0.83	0.83	0.83	0.83	0.83	0.83
Iterative solver	0.02	0.00	0.02	0.02	0.02	0.02
Total solver time	15.11	14.95	14.98	14.95	14.95	14.95

Table 5.5 CPU time distribution of parallel solver in the master processor

Solver stages	*Iteration 1*	*Iteration 2*	*Iteration 3*	*Iteration 4*	*Iteration 5*	*Iteration 6*
Pre-solver	0.02~0.06	0.0	0.0	0.0	0.0	0.0
Assemble K	0.94~1.09	1.05~1.17	1.05~1.09	1.03~1.20	1.03~1.19	1.05~1.24
Forming S_p	8.67~12.56	8.59~12.55	8.56~12.59	8.53~12.55	8.53~12.56	8.67~12.56
Send S_p	0.42~1.77	0.42~1.75	0.41~1.73	0.42~1.75	0.42~1.75	0.41~1.75
Forming y_p	0.02~0.03	0.01~0.02	0.01~0.03	0.0~0.03	0.02~0.03	0.02~0.03
Send y_p	0.0~0.03	0.0	0.0	0.0	0.0	0.0
Receive x_b^p	0.86~4.25	0.86~3.56	0.86~3.62	0.86~3.64	0.86~3.64	0.86~3.47
Internal variable x_i	0.02	0.02~0.03	0.02	0.02	0.0~0.03	0.02
Reaction force	0.0~0.01	0.0	0.0	0.0	0.0	0.0
Total solver time	15.03~15.71	14.98~14.99	14.99~15.02	14.96~14.99	14.96~15.00	14.96~14.99

Table 5.6 CPU time distribution of parallel solver in slave processors

The solution of this example reaches convergence after 6 iterations, the CPU time on the parallel solver stage is about 89.89 second and takes 77.41% of the total analysis time, when 4 processors are used. The CPU time distributions at each iteration, for

each sub-stage of the parallel solver stage for the master and slave processors are given in Tables 5.5 and 5.6. It shows that the most time costs are spent on waiting, receive $\boldsymbol{S}_p$ and preconditioning of the global Schur matrix, which take 73%, 17.47%, 5.49% of the total solver time for the first iteration in the parallel solver for the master processor; the time distributions for the remaining iterations are almost the same. Forming the $\boldsymbol{S}_p$ matrix at the first iteration for each slave processor, as shown in Table 5.6, takes from 8.67 to 12.56 seconds, this is due to a slightly unequal load balance among the slave processors. It can be seen that assembling the $\boldsymbol{K}_p$ matrix, forming and sending the $\boldsymbol{S}_p$ matrix takes about 6.94%, 79.95% and 11.26% of the total solver time on a slave processor. This is the reason why it costs 11.06 seconds waiting time on the master processor. The most expensive operation in the parallel solver stage is at the blocked modified Cholesky factorisation phase on the slave processor, in order to obtain a local Schur complement matrix $\boldsymbol{S}_p$. The costs of assembling the global Schur matrix and the solution of the resulting interfacial equations on the master processor are negligible.

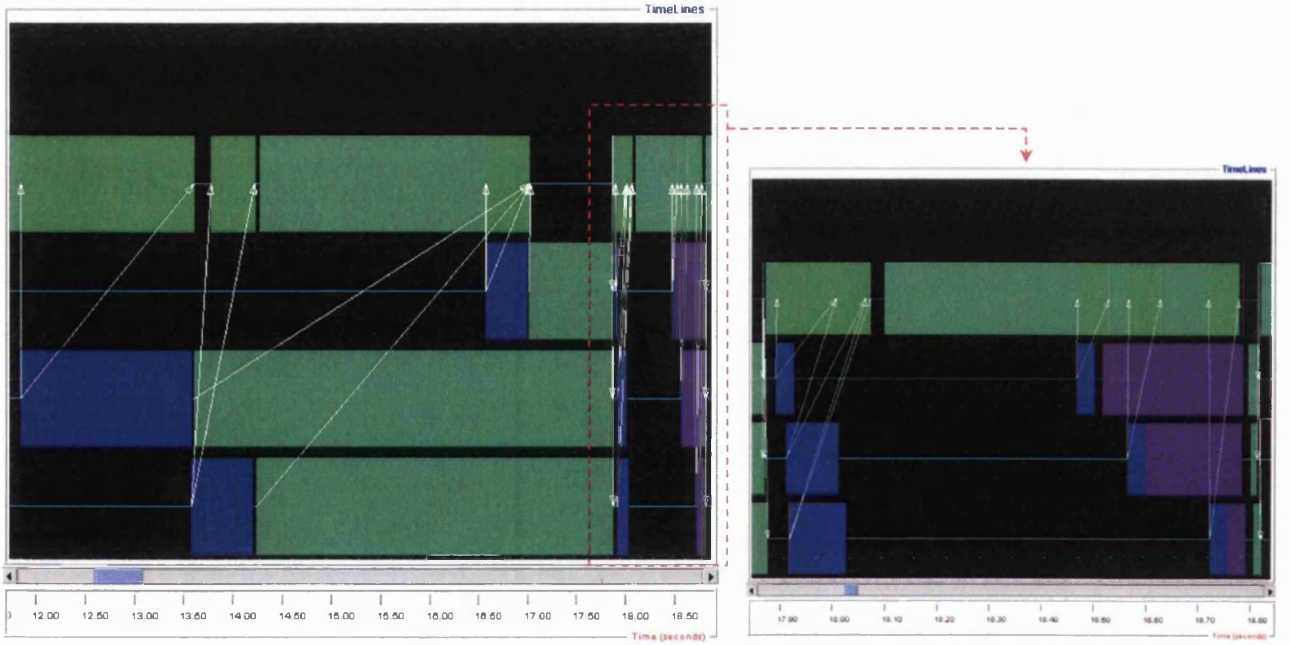


Figure 5.28 (a) Time results at first iteration (b) Time results at post-solution stage

A timeline result generated by the visualization tool *Jumpshot* is shown in Figure 5.28. Figure 5.28(a) illustrates the time spent on computation, sending and receiving records and synchronisation in the solver and post-solution stages of the first iteration. The first line is for the master processor, while the next three lines show the timeline results for the three slave processors. It is obvious that most time is spent on forming and sending the $\boldsymbol{S}_p$ matrix, as shown in Figure 5.28(a), Figure 5.28(b) shows the time spent on the post-solution stage, which includes sending the internal displacement vector, calculating and sending the internal force vector to the master processor for convergence check.

5.4.2 Explicit analysis of a simply supported T beam

The test case set in section 5.4.1 is adopted to assess the performance of explicit parallel analysis. The geometry, boundary conditions and finite element mesh of the T-beam are illustrated in Figure 5.26 and the material properties are given in Table 5.3. An eight-noded hexahedral element with reduced one Gauss integration point is used. The element formulation is based on the assumed strain stabilization method [5.11][5.12], which provides a more robust hourglass control than other 3-D hexahedral elements in explicit codes. A ramped loading curve is defined with $P = 0 \sim 0.2\, N/mm^2$ within the time interval of 0.001 seconds. The time step is chosen as $\Delta t \approx 0.214 \times 10^{-5}$ seconds and the total simulation time is 0.0214 seconds with 10000 time steps. A buffer size for the internal or external zone of each subdomain is taken as 5%-10% of the subdomain length in order to keep the number of overlapped elements to a minimum.

Number of Processors	*CPU Time* (s)	*Speed-up*	*Efficiency*
1	1108.32	-	-
3	327.03	3.389	1.1296
4	223.0	4.97	1.2425
5	200.39	5.53	1.106

Table 5.7 Parallel speed-up and efficiency

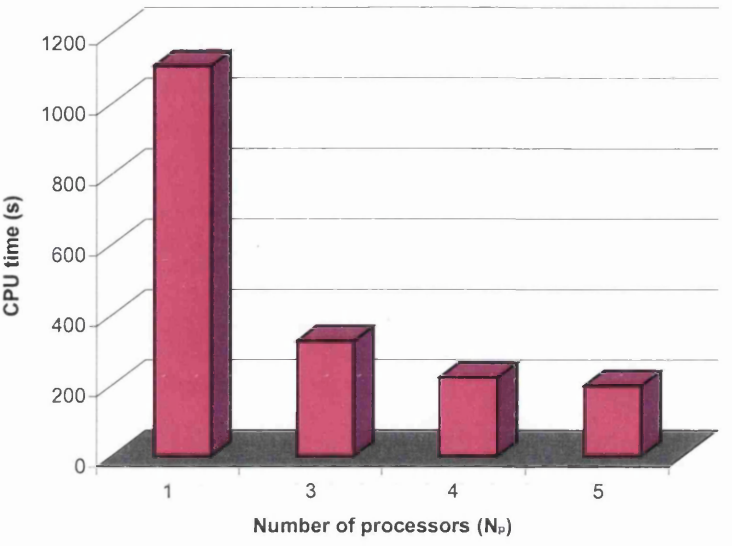


Figure 5.29(a) CPU time versus number of processors

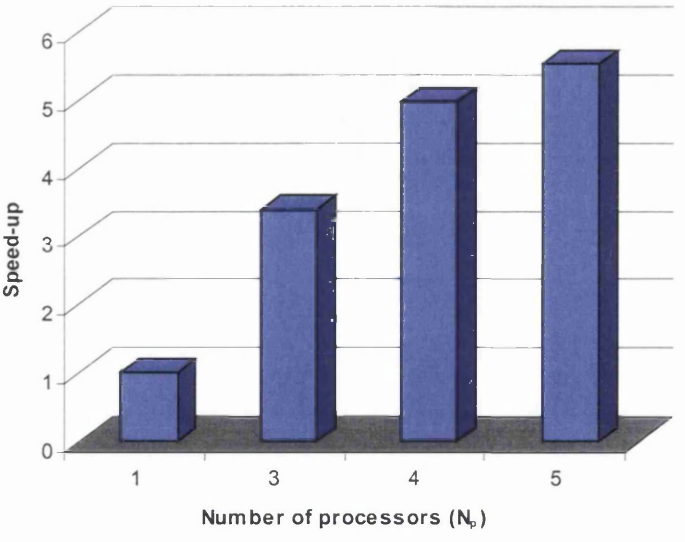


Figure 5.29(b) Parallel speed-up

The explicit parallel test is run on 1~5 processors on a PC interconnection network system and the performance of parallel speed-up and efficiency is given in Table 5.7. It shows superlinear speedup, i.e. $S\left(N_p\right) > N_p$, this is due to significant reduction in

CPU time on the internal force calculation. Figure 5.29(a)(b) illustrate the CPU time taken against the number of processors and parallel speed up. The CPU time distributions at each sub-stage of the explicit parallel analysis are give at Table 5.8 with one, four and five processors, respectively. It shows that the most time costs are spent on load assembly and internal force calculation for a sequential analysis, it takes 103.31 and 928.31 seconds or about 9.32 % and 83.75% of the total CPU time. For a parallel analysis with 5 processors the time spent on load assembly and internal force calculation is reduced to 13.65~22.55 and 87.83~104.81 seconds, taking about 6.84~11.23% and 44.0~52.19% of the total CPU time. Since the external variables on the boundaries are updated using neighbouring interfacial variables at each time step, the time spent on inter-processor communication is unavoidable. It is noted that the solution communication time on the parallel analysis with using 5 processors takes 32.34~46.49 seconds and is slightly increased compared with using 4 processors. The experimental tests on the explicit parallel analysis illustrate that the communication time is linearly proportional to the number of bytes to be sent; about 10 MB in 0.8 seconds.

Solver stages	***Single processor***	***4 processors***	***5 processors***
Time integration	61.3 (5.52%)	9.98~15.93 (4.48~7.14%)	7.98~13.24 (4.0~6.59%)
Solution communication	0.0	27.81~38.22 (12.48~17.13%)	32.34~46.49 (16.35~23.15%)
Load assemble	103.31 (9.32%)	17.19~27.35 (7.72~12.26%)	13.65~22.55 (6.84~11.23%)
Internal force calculation	928.31 (83.75%)	115.42~125.42 (51.82~56.24%)	87.83~104.81 (44.0~52.19%)
Output	11.91 (1.07%)	3.88~5.89 (1.74~2.64%)	3.38~4.84 (1.69~2.41%)
Total time	1108.32	222.7~223.0	199.59~200.81

Table 5.8 CPU time distribution of parallel solver with 4 and 5 processors

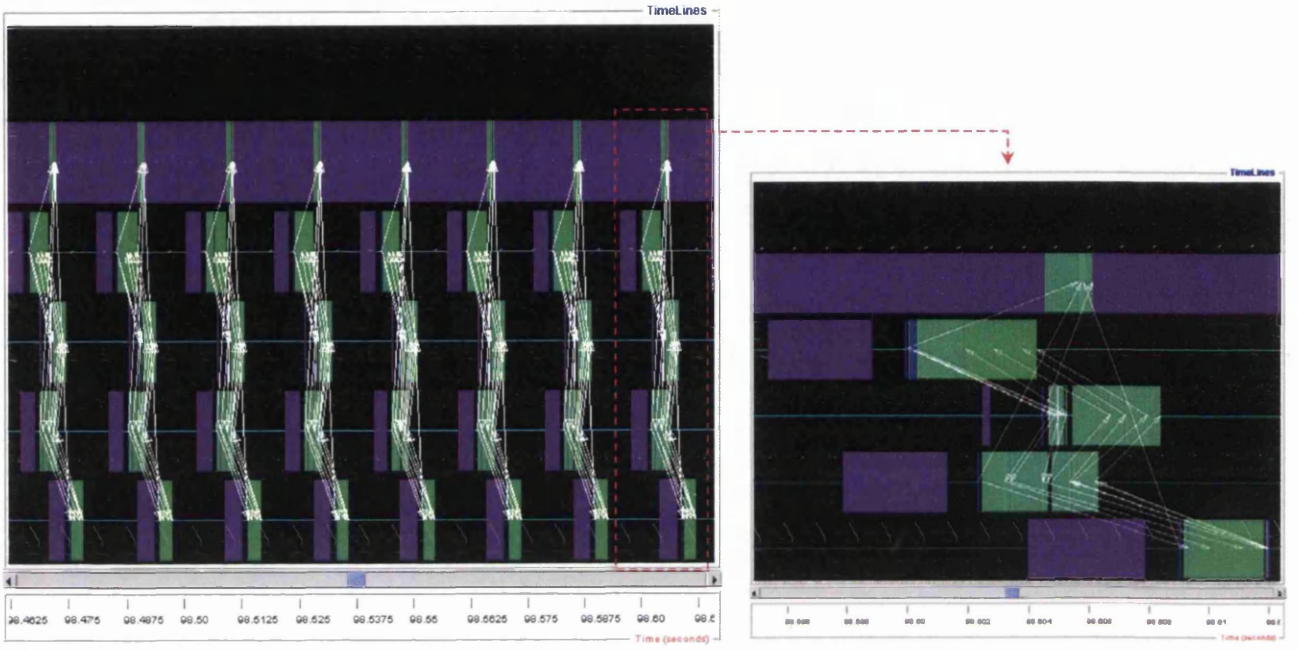


Figure 5.30 Parallel operation of a single time step

The timeline results for a typical time step of the explicit parallel analysis with 5 processors is shown in Figure 5.30, which illustrates the communication between neighbouring slave processors and synchronisation among the master and slave processors. The first line is for the master processor and the next four lines show the timeline results for the four slave processors. It can be seen that the most communication time is spent on sending and receiving interfacial and external variables between the neighbouring slave processors.

5.4.3 Explicit analysis of 3-D horizontal water sloshing

The 2-D explicit finite element modelling of horizontal water sloshing in section 5.1.4 is extended to the 3-D case with a 200 mm width simulating the half width of the water tank. The fluid geometry and the tank wall are discretized into 25482 and 9797 four-noded tetrahedral elements respectively. The tank is assumed to be rigid solid and is applied with the prescribed velocity, which is defined in Figure 5.2. A 3-D node-facet contact algorithm is used to simulate contact interaction between the fluid and the rigid wall, and a frictionless law is applied. The fluid properties are kept the same as the 2-D case and the stabilization constant $\alpha = 0.5\times10^{4}$. A critical time step Δt_{cr} is about 0.135×10^{-6} seconds with total 2000 time steps being simulated. A buffer size for the internal and external zone of each subdomain is chosen at 5-8% of subdomain length according to the number of processors used.

The explicit parallel test is run on 1~5 processors. Figure 5.31(a)(b) shows the global mesh and nodal status on the subdomain mesh of each processor (using four slave processors and 8% buffer size). In Figure 5.31(b) the nodes with red colour denote the external nodes, which are overlapping with the neighbouring subdomain. The nodes with green colour represent the interfacial nodes. Table 5.9 and Figure 5.32 present the performance of speed-up and efficiency of the parallel analysis. It indicates that the explicit parallel analysis gives a high rate of efficiency. Again, the superlinear speedup is obtained, since the CPU times on the internal and contact force calculation are reduced significantly.

Number of Processors	*Time* (s)	*Speed-up*	*Efficiency*
1	978.28	-	-
3	290.73	3.354	1.118
4	219.53	4.442	1.111
5	168.67	5.781	1.156

Table 5.9 Parallel speed-up and efficiency

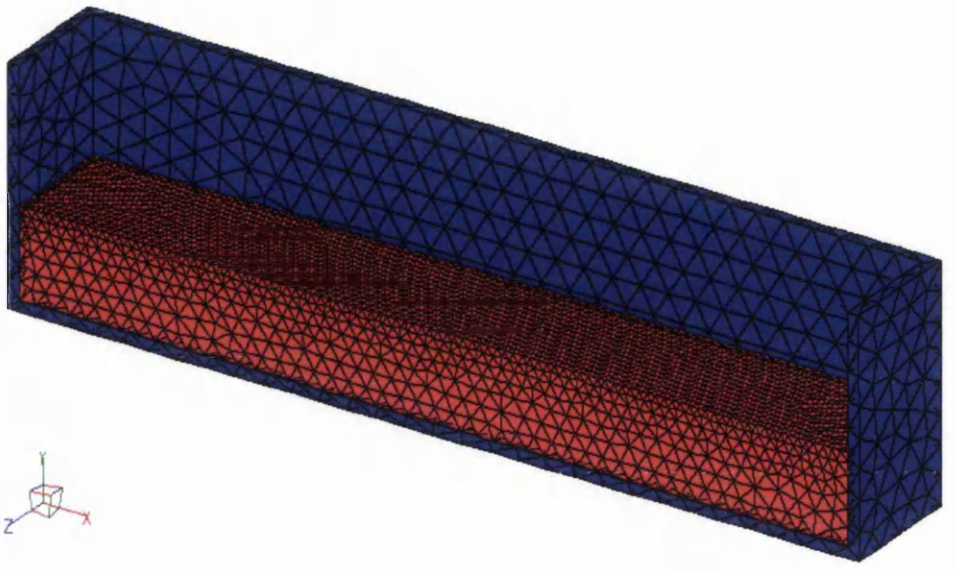


Figure 5.31(a) Global mesh

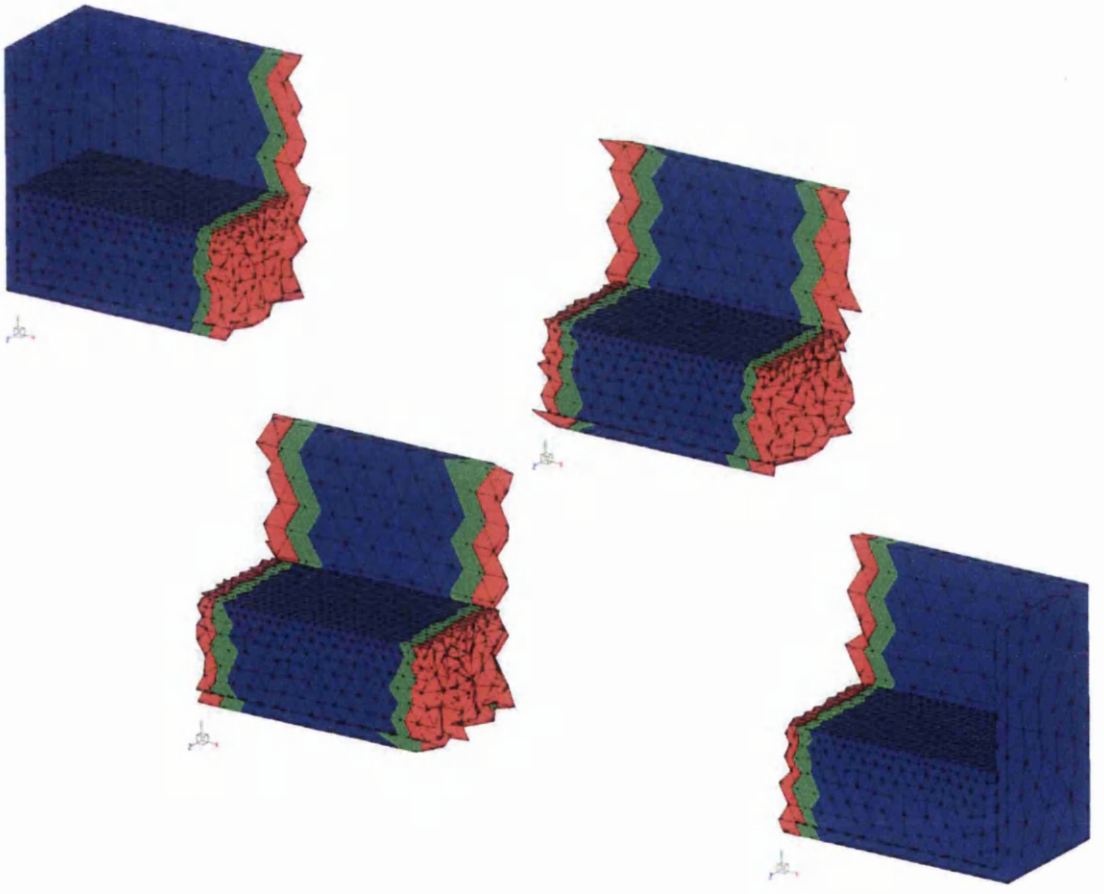


Figure 5.31(b) Partitioned parallel nodal status

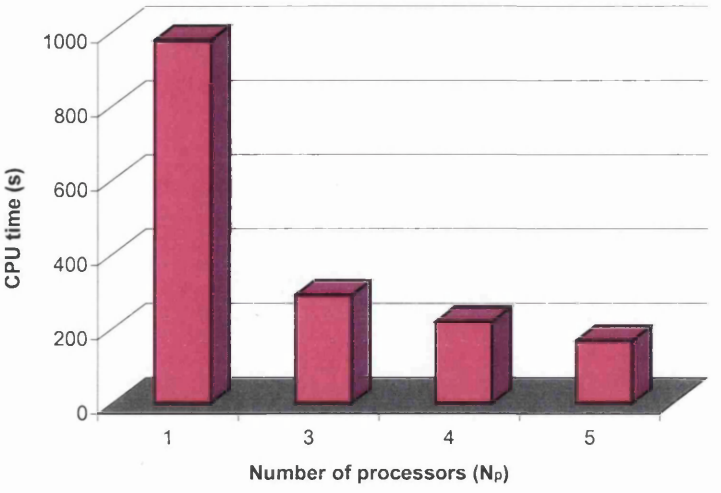


Figure 5.32(a) CPU time versus number of processors

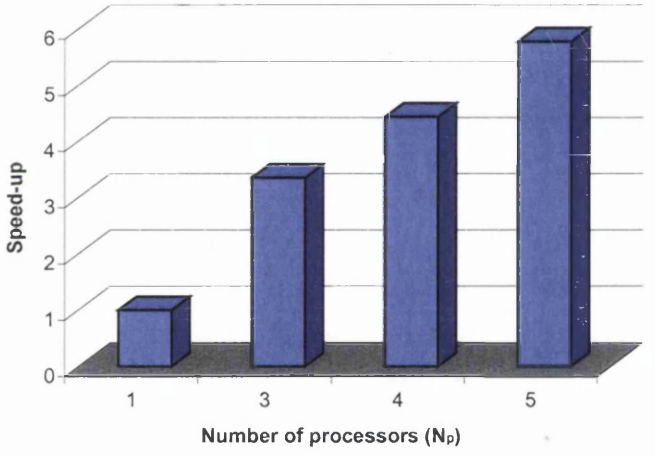


Figure 5.32(b) Parallel speed-up

The CPU time distributions at each sub-stage of the explicit 3-D water sloshing analysis are given in Table 5.10, with one, four and five processors used. It shows that the most time costs are spent on internal force and contact force calculation for a sequential analysis, which takes 536.73 and 406.73 seconds, i.e. up to 96.3% of the total CPU time. For a parallel analysis with 5 processors the time spent on internal force and contact force calculation are reduced to 67.96~74.23 and 64.03~70.97 seconds, respectively, consuming about 78.66~85.79% of the total CPU time. In this

test case the time spent on the solution communication is very little, only taking 4.29~5.2% of the total CPU time.

Solver stages	*Single processor*	*4 processors*	*5 processors*
Equation solver	13.56 (1.38%)	2.38~4.03 (1.09~1.83%)	1.88~3.36 (1.12~1.98%)
Solution communication	0.0	6.5~9.44 (3.0~4.3%)	7.20~8.81 (4.29~5.2%)
Load assemble	9.94 (1.01%)	1.92~3.91 (0.88~1.78%)	1.87~3.36 (1.11~1.99%)
Internal force calculation	536.73 (54.86%)	88.52~101.02 (40.76~46.0%)	67.96~74.23 (40.5~43.86%)
Contact force calculation	406.73 (41.57%)	84.18~91.95 (38.76~41.87%)	64.03~70.97 (38.16~41.93%)
Output	8.97 (0.92%)	3.62~6.33 (1.67~2.88%)	2.71~3.96 (1.6~2.34%)
Total time	978.28	217.17~219.6	167.79~169.24

Table 5.10 CPU time distribution of the parallel solver with 4 and 5 processors

5.5 References

[5.1] M.S. Turnbull, A.G.L. Borthwick, and R. Eatock Taylor. Numerical wave tank based on a σ-transformed finite element inviscid flow solver. *Int. J. Numer. Meth. Fluids*. 42 (2003) 641-663.

[5.2] H. Bredmose, M. Brocchini, D. H Peregrine, and L. Thais. Experimental investigation and numerical modelling of steep forced water waves. *J. Fluid. Mech.* 490 (2003) 217-249.

[5.3] G. Wei, J.T. Kirby, S.T. Grilli and R. Subramanya, A fully nonlinear Boussinesq model for surface waves. Part 1. Highly nonlinear unsteady waves. *J. Fluid. Mech.* 294 (1995) 71-92.

[5.4] P. Hansbo, Lagrangian incompressible flow computations in three dimensions by use of space-time finite elements. *Int. J. Num. Meth. Fluids*. 20 (1995) 989-1001.

[5.5] P. Hansbo, The characteristics streamline diffusion method for the time-dependent incompressible Navier-Stokes equations. *Comput. Meth. Appl. Mech. Engng*. 99 (1992) 171-186.

[5.6] Parallel processing – using the packaging approach, internal report IN_1356_packaging, Rockfield Software Ltd. (2003)

[5.7] William Group, Ewing Lusk, and Anthony Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface. The MIT Press, Massachusetts (1994)

[5.8] William Group, Ewing Lusk, and Rajjev Thakur, Using MPI-2:Advanced Features of the Message-Passing Interface. The MIT Press, Cambridge, Massachusetts (1999)

[5.9] E.A. de Souza Neto, D. Peric and D.R.J. Owen, Design of a simple low order finite elements for larger strain analysis of nearly incompressible solids. *Int. J. Solids. Struct.* Vol 33. (1996) 3277-3296.

[5.10] W. Liu, Finite element analysis for large incompressible deformations. CM/386/01. School of Engineering, University of Wales Swansea. (2001)

[5.11] T. Belytschko and L.P. Bindeman, Assumed strain stabilization of the eight node hexahedral element. *Comput. Meth. Appl. Mech. Engng.* 105 (1993) 225-260.

[5.12] T. Belytschko and L.P. Bindeman, Assumed strain stabilization of the 4-node quadrilateral with 1-point quadrature for nonlinear problems. *Comput. Meth. Appl. Mech. Engng.* 88 (1991) 311-340.

Chapter 6

Conclusions

6.1 Summary and conclusions

The main motivation of the research herein is the development of rational parallel computational strategies for solving an incompressible or slightly compressible transient Stokes flow with moving boundaries and interfaces. A brief summary of the principal field of the research is presented together with a set of open-ended conclusions in the following.

6.1.1 Lagrangian formulation for transient Stokes flow

A comprehensive strategy for the solution of incompressible or slightly compressible transient Stokes flow problems with moving boundaries has been presented and implemented. It includes a novel space-time Galerkin/least-square finite element technique in the Lagrangian frame, which comprises the least square terms in the variational formulation and prevents numerical oscillation on the pressure field. It is evident that the stabilization constant α in the formulation can be chosen within a wide range of values. A stable solution can be obtained by using lower equal order interpolation functions for velocity and pressure fields without violating the Babuška-Brezzi stability condition.

The space-time Galerkin/least-square finite element formulation has also been successfully extended to the explicit analysis. The forward Euler method is used in the time integration procedure and by controlling a critical time step the conditional stability of the solution can be achieved. The formulation has been proved to be a very attractive tool for simulation of fluid-structure or fluid-fluid particle interaction problems.

6.1.2 Contact modelling and adaptive remeshing

Three major contact algorithms for enforcing the contact constraints have been reviewed in the dissertation. The penalty method based discrete element contact algorithm, 2-D and 3-D node to facet algorithm, is adopted to simulate fluid-structure or fluid-fluid particle contact. In the explicit modelling of liquid sloshing examples, the frictionless contact simulation gives the best prediction of the wave propagation, in terms of forced wave amplitude and period. The Coulomb friction law is more suitable for fluid-fluid particle contact for the viscous fluids, which was shown in the Shampoo filling example.

The adaptive remeshing technique has been used to deal with large deformation of Lagrangian meshes. It improves the accuracy of the finite element solution and enables it to carry on the simulation by overcoming excessive element distortions. New mesh density prediction is calculated according to *a posteriori* error estimator, which is based on the velocity gradient error norm for the transient Stokes flow. At the field values mapping stage, the weighted least-square mapping algorithm implemented significantly enhances the mapping quality, compared with the background element-mapping scheme.

6.1.3 Implementation of the implicit parallel solver

A hybrid iterative direct parallel solver is implemented in the ELFEN/implicit commercial code. The solver is based on a non-overlapping domain decomposition and sub-structure approach; the solution of the subdomain problem is naturally

parallelized and a modified Cholesky factorisation is used to eliminate the unknown variables of internal nodes at each subdomain. The resulting interfacial equations, which are related with the unknown variables of the interfacial nodes, are solved by a Krylov subspace iterative method. The hybrid iterative direct parallel solver is tested on a PC based interconnection network system and its performance is judged by two measurements, parallel speed-up and efficiency. The performance of the parallel solver is governed by the blocked modified Cholesky factorisation phase in the slave processors and communication of the local Schur matrices. All the results indicate that it does not suffer seriously from the serialization of the backward solution phase in the slave processors and solution of the resulting interfacial equations in the master processor.

6.1.4 Implementation of the explicit parallel solver

The parallelization of explicit finite element fluid dynamics is based on the overlapping domain decomposition and the Schwarz alternating procedure. Due to the dual nature of the overlapping partitioning of the domain a buffer zone between any two adjacent subdomains is introduced for handling the inter-processor communication and updating unknown variables of external nodes on the external zone. Communications for the nodal velocities and pressure are limited to take place only between any two adjacent subdomains, which significantly reduces communication cost. New classification of elements and nodes makes the contact treatment very simple and flexible.

6.1.5 Applications

A number of numerical examples are presented and compared with relevant experimental test cases.

- *2-D and 3-D implicit finite element modelling of horizontal and vertical water sloshing* – which demonstrate that the finite element fluid implicit analysis can

correctly and accurately depict wave propagations within the tank that is excited by a prescribed motion.

- *2-D explicit modelling of liquid sloshing within a water tank and a fragrance bottle* – which validates the explicit fluid dynamics formulation developed in this work. The contact between fluid and the rigid tank wall is applied by using a 2-D node-to-facet discrete contact algorithm and a frictionless contact model.

- *Simulations of collapse of a 2-D axisymmetric liquid column and a 3-D liquid column* – which also provide an assessment of the performance of the schemes proposed in this work.

- *2-D explicit modelling of shampoo container filling* – which involves complicated contact interaction between fluid and the rigid container walls.

- *A simply supported 3-D T-beam test* – which evaluates the parallel performance of the implicit and explicit parallel solver implemented.

- *Explicit analysis of 3-D horizontal water sloshing test* – it evaluates the parallel performance of the explicit parallel solver, which involves discrete element contact.

6.2 Recommendations for further work

A number of fundamental issues, which require to be improved in future work, are addressed here. In the following, possible extensions to the main topics of research of this thesis are presented.

6.2.1 Domain partitioning

When implementing a domain decomposition strategy on a parallel computing system, efficient techniques must be available for partitioning an arbitrary graph. In the present research work, the partitioning method is based on a simple algorithm, in which a domain is split along its most elongated coordinate direction. Obviously this is a special situation. Most static and dynamic partitioners used for domain decomposition can be catalogued into two classes, geometric and topological [6.1][6.2]. The geometric approach works on the physical mesh and requires the coordinates of the mesh points to find adequate partitioning. The method is well suited to problems in which interactions are inherently geometric, such as particle simulation or contact detection. One example is the Recursive Coordinate Bisection (RCB) algorithm, which was proposed by Berger and Bokhari [6.3]. Recently, a modified RCB method was introduced by Wang *et al* [6.4]. It can be implemented in the explicit parallel solver, but the number of subdomains can only be a power of two, i.e. 2, 4, 8, 16 and so on. Each subdomain is a simple rectangular parallelepiped, since the cutting planes are confined to be orthogonal to an axis.

The topological method works with the connectivity information of the elements in a mesh, instead of geometric coordinates. The connectivity is generally described as a graph. The methods are best suited to partitioning computational meshes, where the connectivity is implicit in the mesh. One of the most popular topological methods is known as Recursive Spectral Bisection (RSB) [6.5]. METIS[6.6] and its MPI implementation ParMETIS[6.7], both based on the topological method, are public domain software packages for partitioning general graphs. The interface to those partitioning packages may produce the required quality partitions for any complex graph.

6.2.2 Adaptive remeshing with parallelisation

The adaptive remeshing technique has been only applied in the sequential analysis in this work, since the reduction of computational cost for adaptive remeshing on parallel analysis requires further work. This is due to: (a) an automatic mesh generator

is normally a separate program and works sequentially; (b) Partitioning of domains for a newly generated mesh is also a sequential process, which requires a significant computational time for a complex geometry.

6.3 References

[6.1] B. Hendrickson and K. Devine, Dynamic load balancing in computational mechanics, Comput. Methods Appl. Mech. Engrg. 184 (2000) 485-500.

[6.2] S.H. Hsien, G.H. Paulino and J.F. Abel, Evaluation of automatic domain partitioning algorithm for parallel finite element analysis. International J. for Numerical Methods in Engrg., 40, (1997) 1025-1051.

[6.3] M.J. Berger and S.H. Bokhari, A Partitioning Strategy for Non-uniform Problems on Multiprocessors, IEEE Trans. Computers, C 36 (1987) 570-580.

[6.4] F. Wang, Y.T. Feng and D.R.J. Owen, Parallelisation for finite-discrete element analysis on distributed-memory environment, Report of INME, University College of Swansea.

[6.5] A.Pothen, H.D. Simon and K. Liou, Partitioning sparse matrices with eigenvectors of graphs, SIAM J. Matrix Anal. Appl., Vol. 11 No. 3 (1990) 430-452

[6.6] G.Karypis and V.Kumar, METIS 4.0: Unstructured graph partitioning and sparse matrix ordering system. Technical report, Department of Computer Science, University of Minneota, (1998).

[6.7] G.Karypis, K.Schloegel and V.Kumar, ParMETIS 2.0: Parallel graph partitioning and sparse matrix ordering library. Technical report, Department of Computer Science, University of Minneota, (1998).