An adaptive Space-Time Boundary Element Method for Impulsive Wave Propagation in Elastodynamics

Joseph Xu Zhou

A Thesis submitted for the degree of
Doctor of Philosophy

UNIVERSITY
of
GLASGOW

Department of Civil Engineering
University of Glasgow
April 2007

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Dedicated to

my wife Sharon and my parents

Abstract

Wave propagation in natural or man-made bodies is an important problem in civil engineering, electronic engineering and ocean engineering etc. Common examples of wave problems include earthquake wave modeling, ocean wave modeling, soil-structure interaction, geological prospecting, and acoustic or radio wave diffraction. The Boundary Element Method (BEM) is a widely-used numerical method to solve such problems in both science and engineering fields.

However, conventional BEM modeling of wave problems encounters many difficulties. Firstly, the method is expensive since influence matrices are computed at each time step and BEM solutions at every former time step have to be stored. Secondly, if large time steps are used, inaccuracies arise in BEM solutions; but if small time steps are used, computational costs become impractical. Thirdly, the dimensionless space-time ratio must be limited to a narrow range to produce a stable solution.

In this thesis, we attack these problems by introducing adaptive schemes and mesh refinement. Instead of using uniform meshes and uniform time steps, error indicators are employed to locate high-gradient areas; then mesh refinement in space-time is used to improve the resolution in those areas only. Another strategy is to introduce the space-time concept to track moving wave fronts. In wave problems, wave fronts move in space-time, and high gradients arise both in space and in time. It is thus inadequate to refine the mesh in space only because there are high gradients in time as well. Hence, besides a locally mesh refinement scheme employed in space, local time stepping is also used to improve the accuracy and efficiency of the algorithm.

This adaptive scheme is implemented in the *C* language and used to solve scalar and elastodynamic 2D and 3D wave propagation problems in a open and closed field. Gradient-based and resolution-based error indicators are employed to locate these moving high-gradient areas. A space mesh refinement scheme and the local time

stepping is used to refine the area to achieve higher accuracy. The adaptive BEM solver is $1.4 \sim 1.8$ times faster than the conventional BEM solver. It is also more stable than the conventional BEM. We also parallelize the BEM solver to further improve its efficiency. Compared with the non-parallel code, using a 8-processor Linux cluster, a speed-up factor of four is achieved. This suggests that substantial further gains can be obtained if a larger parallel computer is available.

Declaration

The work in this thesis is based on research carried out at the Department of Civil Engineering, University of Glasgow. No part of this thesis has been submitted elsewhere for any other degree or qualification and it all my own work unless referenced to the contrary in the text.

Acknowledgments

My years at Glasgow really got me started in the academic career. I have never learned so much and I am convinced that I will dedicate my whole life to science. Every morning when I was jogging in the beautiful Kelvin Grove park, which is full of trees, grass and flowers, I can not help feeling how lucky I am. After working in software industry for three full years, I deeply appreciate that now I enter a free society of scholars, who are animated by intellectual purposes and are left to pursue their own ends in their own way without distraction of worldly concerns. The life here is really like a *Shangri-la* to me. I am so fortunate to have three quiet years to sit down to read many books of mathematics, physics and mechanics etc. which I am interested, and have time to contemplate some problems and try hard to find solutions for them. I also have a group of friendly and helpful people around me. I definitely would not have been who I am today without the help of them.

First of all, I would like to thank my advisor, Dr. Trevor G. Davies. I am very grateful to him for giving me the opportunity to finish my PhD under his supervision. Dr. Davies belongs to the group of researchers who are really motivated by their curiosities to know and happiness when they understand (I still remember that how happy he was after attending an illuminating seminar about the sundial). I learned a great deal from him on how to be a good researcher. I would also like to thank Prof. Nenad Bicanic for being my internal examiner. When I finished my first-year research report, it was Nenad who pointed out to me how to write a report more consistently and more logically. I also appreciate that Nenad invited many researchers from different fields to give interesting seminars, which I always like to go and learn a lot from them. Nenad is also an universal man for all sports, and he never spared any effort to spread sports enthusiasm to all students, whether

they are undergraduates or PhDs. I will always remember how he always almost single-handed organizes the yearly sports event, which motivated us PhD students to practice basketball every Friday.

I also thank Prof. G. D. Manolis for his advice, and I learned a lot about critical thinking in science from him during an international conference. It was very helpful for him to share his BEM codes with me, which benchmarks mine and gave me confidence in my new code. Also, Dr. Jon Trevelyan discussed with me the critical dimensionless ratio, which shed light on some problems I solved later.

My life at Glasgow would not have been so fruitful without many good colleagues and friends here. Tomasz Koziara is a nice person to collaborate with: we did a study on fast space search to accelerate the computational speed of dynamic BEM, which led to a joint paper published in 2006. I also thank Ken McColl for helping me out with computer problems.

There are other people who have helped me greatly in the last several years. Eileen Davies has provided much help to me in arranging my participation in conferences and working as a demonstrator. Ms Pat Duncan, Secretary of the Engineering Faculty, helped me when I had registration problems. Barbara Grant, the secretary of the department, also has provided much help during the past three years.

I have reserved my final thanks for my families. My parents have given me all their love over the years. My sister Li Zhou has supported my application for studying abroad. My parents-in-law have loved me like their own son. Finally, my deepest thanks go to my wife, Sharon Hong Zhang, for supporting my study, shouldering all the labor to decorate our new flat and bearing all the difficulties of being pregnant when I was not around.

July 11. 2007

Contents

List of Figures

List of Tables

Nomenclature

τ	time
$H(x)$	Heaviside function
q^*_{2D}	Traction Green function of 2D wave equation
t_0	time of source point
u^*_{2D}	Displacement Green function of 2D wave equation
$v_0\,(\mathbf{x})$	velocities on the boundary when time t=0
(s_1, s_2, n)	general curved orthonormal basis
$(x_1^{'}, x_2^{'}, x_3^{'})$	a local Cartesian coordinate system
$\bar{\eta}^q_G$	principal global traction error tolerances
$\bar{\eta}^u_G$	principal global displacement error tolerances
β	space-time ratio defined as $\beta = \frac{c \cdot \Delta t}{\Delta h}$
$\bigtriangledown$	differential operator
$\bigtriangledown^2$	Laplace operator
$\delta(x)$	Dirac delta function
η^q	the global error tractions
η^u	the global error displacements
Γ	Boundary of the problem domain

Γ_1 Boundary on which the potential/displacements are prescribed

Γ_2 Boundary on which the flux/tractions are prescribed

Γ_k kth boundary element

$u_{,ii}$ tensor notation for Laplace operator

$\mathbf{b}$ body force

$\mathbf{n}$ unit normal vector

$\mathbf{p}(\mathbf{x}, t)$ tractions on the boundary

$\mathbf{x}$ the field point

$\mathbf{x}_0$ the source point

μ, λ Lame's constants

ν Poisson's ratio, its relation with Lame's constants is $\mu = \frac{E}{2(1+v)}$; $\lambda = \frac{E}{(1+\nu)(1-2\nu)}$

Ω Volume of the problem domain

$\phi(x, y, z)$ twice-differentiable functions

$\phi_i(\xi)$ shape function

$\Phi_m(t)$ piece-wise linear shape function in time

$\sigma_{\alpha\beta}$ stress tensor

σ_{km} stress tensor

σ_{lkm} represent components of the stress tensor in the k and m direction when a point load is applied in the l direction

σ^*_{lkm} Displacement Green function of 3D Elastodynamics equation

$\mathbf{G}^{(nn)ij}_{lk}$ influnce matrix from displacement Green function in 3D Elastodynamics

$\mathbf{H}_{lk}^{(nn)ij}$ influnce matrix from traction Green function in 3D Elastodynamics

$\mathbf{u}(\mathbf{x}, t)$ displacement in 4D space-time

$\varepsilon_{\alpha\beta}$ strain tensor

$\triangle\, h$ mesh size

$\triangle\, t$ time step

C contant in the inequalities

c wave speed

C_1 pressure wave speed

C_2 shear wave speed

c_i jump term due to the singularity of the source point

E Young's modulus

e_σ error indicator for boundary stresses

e_q point-wise errors of tractions

e_u point-wise errors of displacements

f frequnce of the vibration

$F(t)$ External force

$f(x, y, z)$ twice-differentiable functions

$G(\mathbf{x}, t, \mathbf{x}_0)$ Green function

$G^D(\mathbf{x}, t, \mathbf{x}_0)$ Green function dipole

G_{ij}^{nm} influnce matrix from displacement Green function

Gid a software for pre-processing data

H_{ij}^{nm} influnce matrix from traction Green function

$J(\xi)$ Jacobian

L length of the element in 1D

$N^{mj}(\xi,\tau)$ two-dimensional shape function in space-time

$P(t)$ Presure variation of the time

P_m spatial variation of the traction at time step m

p^*_{lkm} Traction Green function of 3D Elastodynamics equation

r distance in space-time

$R^N(\tau$ the algorithm for advancing classes N in time over the time interval τ

r_{max} radius of the big sphere

r_{min} radius of the small sphere

S Boundary surface in 3D problems

T^+ half open time interval $[0,\infty)$

t^n nth time step

t_0^+ means to take the upper limit of time t_0 to avoid boundary integral falling exactly on the time t_0

tol an error tolerance given beforehand

$u_0(\mathbf{x})$ displacements on the boundary when time t=0

Chapter 1

Introduction

1.1 Background of research

Wave propagation in natural or man-made bodies is an important problem in civil engineering, electronic engineering and ocean engineering etc. Common examples of wave problems include earthquake wave modeling, ocean wave modeling, soil-structure interaction, geological prospecting, and acoustic or radio wave diffraction.

The mathematical partial differential equation model for linear wave propagation and scattering was developed as early as 1747 (by the French mathematician, *D'Alembert*). However, since solving a PDE with complicated boundary conditions is never a trivial task, wave problems are normally solved numerically. Numerical modeling techniques for wave equations have been developed since the advent of computers from the 1960s. These may be categorized into domain-type numerical methods such as finite difference methods (FDM), and boundary-type ones such as boundary integral equation methods (BEM). The difference between a domain mesh and a boundary mesh is shown in Fig. 1.1.

Compared with domain-type numerical methods such as FDM or FEM, boundary-type methods have several advantages. Firstly, the problem scale in BEM is much smaller since only boundaries are discretized and only boundary physical quantities are solved. Secondly, the BEM solution automatically satisfies the infinite boundary conditions in an open domain. Thirdly, the notorious problem of numerical damping in the wave propagation problem is less severe since the BEM model employs fewer

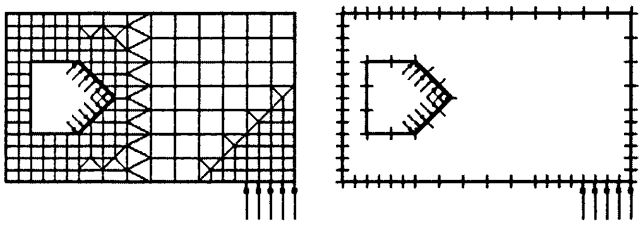


Figure 1.1: A domain mesh and a boundary mesh (After Prof. Lothar Gaul, Universität Stuttgart, Germany [28])

nodes. Finally, the BEM provides efficient representation of cracks and geological features, such as faults and planes, making it a powerful analysis tool for crack wave scattering problems. Hence, BEM has become a widely-used numerical methods to solve wave problems in both science and engineering fields.

The first time-domain Boundary Element formulation was introduced by Cole (1978) [15] to solve transient 2D elastodynamic problems for the anti-plane case. A general boundary element scheme to solve the 2D transient elastodynamics problem was derived by Mansur & Brebbia (1983) [54]. Later, similar methods for 2D scalar wave and elastodynamic wave propagation were also developed by Banerjee & Ahmad (1988) [2]. The 3D elastodynamic time-domain BEM formulation and implementation were studied by Karabalis & Beskos (1984, 1985) [43] in the context of 3D dynamic soil-structure interaction problems. However, these early research works lacked a systematic analysis of error estimation, convergence and stability. Instabilities were observed even for uniform meshes when the dimensionless parameter, space-time ratio β (the ratio between time steps and element size), fell outside a specific range $[0.3 \sim 1.5]$.

Over the last decade, there have been increasing efforts to develop more efficient solution techniques. Various spatial and temporal interpolation schemes have been implemented to improve accuracy and efficiency, such as combinations of constant, linear and quadratic functions (Dominguez 1993 [20]), B-splines interpolation schemes (Rizos & Karabalis 1994 [71]), quadratic time interpolation schemes (Wang & Wang 1996 [86]). However, these high-order BEM schemes don't improve the stability of the method. Siebrits& Peirce (1995) [64] have discussed the stability properties of a time domain BEM approach, and have proposed the so-called *half-step* scheme to improve the stability. A time and space weighted method was

also suggested (Yu & Mansur, 1998 [89]), in which the Galerkin method in the time domain was used to make the numerical methods more stable. However, these methods are not necessarily more accurate, because they only improve the stability of BEM by averaging out the instability.

Further difficulties arise when impulse wave propagation problems are solved by numerical methods such as FEM or BEM. Zienkiewicz classes the impulse wave propagation problem in the list of unsolved problems by numerical methods. (Zienkiewicz, 2000 [92]). Because wave phenomena require spatial modeling which is capable of resolving the oscillations, good accuracy cannot be achieved unless at least four quadratic elements (8-10 nodes) are used per wavelength. This places a limitation on the frequency of waves which can be adequately treated on a given mesh. As an example, the scattering of 3 meter radar waves by an aircraft is shown in Fig. 1.2. The solution of the problem required some 15×10^6 elements or approximately 3×10^6 nodes. This size of problem clearly demands substantial computational facilities. However, a 1/10th reduction of the incident wavelength will increase the number of nodes by 10^3 for FEM and about 10^2 for BEM, which is beyond the capacity of very large computers used today, and yet such a wavelength is not extraordinary. Thus, Zienkiewicz drew the conclusion that this problem remains unsolved and a completely new method of approximation is needed to deal with such problems .

In summary, the BEM modeling of wave problems encounters many difficulties. Firstly, solving impulse wave problems in large and complex domains reveals well-known weaknesses in dynamic BEM wave solvers. The solver requires a very fine mesh to accurately represent the moving wave front. The time step also needs to be small enough to correctly represent the sharp gradients in impulse waves. Fine space meshes and small time steps demand large computer memory and long computational time. Thus, there is a need to develop a BEM solver with less memory limitations and higher computational efficiency.

Secondly, it is still difficult for time-domain BEM solvers to produce stable results, specially for impulsive loads and complex geometries. The time domain BEM solver should be made more robust to capture impulsive transient dynamic responses

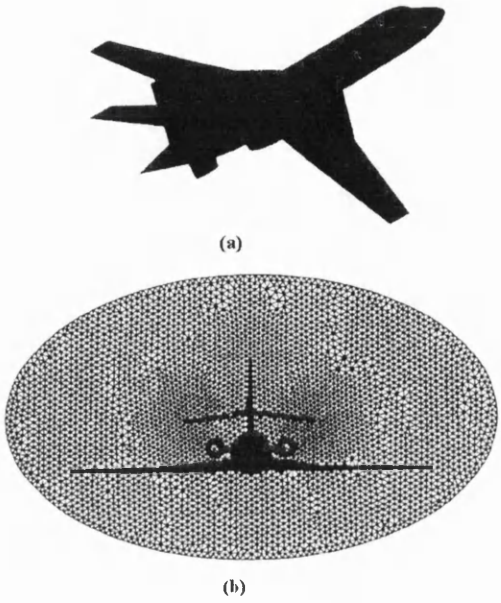


Figure 1.2: (a) the problem domain (b) mesh intersection with aircraft (After Prof. Zienkiewicz, University of Swansea [92])

in any arbitrarily-shaped domain.

Thirdly, the time-stepping algorithm is problematic in wave problems. Because current numerical methods put a strict constraint on the ratio between time step and element size. i.e. the space-time ratio β is constrained within a specific range $[0.3 \sim 1.5]$. Hence, it is still a open problem how to choose the best time marching scheme for BEM with a non-uniform mesh, and how to derive a generally stable time marching scheme. The strategy to solve these problems will be addressed in the next section.

1.2 Research strategies and Objectives

1.2.1 Adaptive strategy

The key issue to improve the computational efficiency of dynamic BEM is to introduce adaptive schemes, which include error estimation and automatic mesh refinements.

If a moving wavefront can not be resolved with a mesh fine enough, all numerical methods, including FEM, FDM and BEM, will produce inaccurate and unstable

results. However, the problem is that because the location of moving wavefronts are not known *a priori*, it can not be predicted where best to introduce mesh refinement. Hence, an adaptive strategy is employed here: computational results from the last time step are postprocessed to locate moving wavefronts; then meshes there are refined, and the problem is solved again. The whole process is repeated until the error tolerance satisfied. The refinement process can be simply to add more nodes near the wavefront, i.e., to use smaller elements to better approximate the function where it changes rapidly.

1.2.2 Space-time concept and Local time stepping

The space-time separation strategy is widely used in solving dynamic problems numerically. The conventional BEM solution follows the same routine: it consists of discretization in space using BEM, and discretization in time using the finite differential method (FDM). However, a difficulty with this strategy arises because the stability condition demands use of small time steps, which are related to the smallest elements in the mesh. Thus, small time steps have to be applied to elements for all regions, which is computationally wasteful. It is more efficient to build a BEM model with local time stepping, combined with a space adaptive scheme which indicates where it should apply. Ideally, in this adaptive scheme, elements are refined and the time step size is adjusted near wave fronts, while other elements remains unaltered where the solution is smooth or quiescent.

In local time stepping, the same time step is no longer applied for all elements. Instead, for impulse wave problems, regions of high stress or high strain will have their own time steps which are different from the rest of mesh. The set of elements are partitioned into N classes according to their different sizes. The bigger time steps will be applied to elements with bigger size, the smaller time steps will be applied to smaller ones. The time stepping will rotate among different classes until all of them reach the next time step.

1.2.3 Research objectives

The aim of the research carried out for this thesis is to develop a framework for an adaptive dynamic BEM solver, which can capture moving wavefronts and refine the mesh around them; offer different local time stepping schemes for elements of different sizes to ensure the accuracy of the solution, and ultimately solve the impulsive wave propagation problem in a more efficient, more accurate and more stable way. The detailed research objectives include:

- An effective *posteriori* error estimation will be developed to locate moving wavefronts by postprocessing numerical results from the previous computation. The error estimation should be computationally cheap, easy to implement, and accurate to identify elements for refinement.
- After the problem elements are identified, automatic mesh refinement will be developed to create a series of conforming elements in smaller sizes than the original one, and to change the mesh topology accordingly.
- A local time stepping scheme will be constructed to offer a range of flexible time steps for elements with different sizes while maintaining time compatibility and stability of the solution.
- A adaptive dynamic BEM programme written in the C language will be developed to implement the error estimation; the automatic mesh refinement and local time stepping schemes. A new BEM solver will also be designed for the adaptive mesh, which will change in each time step.
- A parallel dynamic BEM solver based on Linux clusters will be developed to further improve the efficiency of the BEM solver for impulsive wave problems.

1.3 Outline of Thesis

This thesis consists of eight Chapters, through which the development of the research work is presented from the basic dynamic BEM theory to detailed implementation for 2D wave propagation, 3D wave propagation and 3D elastodynamics problems.

The fundamental mathematical theories are described first to lay down a sound foundation; then the basic strategies and implementations for error estimation, automatic mesh refinement, and adaptive BEM solvers are described in detail with the increasing complexity. The content of each Chapter is summarized below:

In Chapter 2, the literature is reviewed in order to describe the history, *state-of-the-art* and remaining difficulties in solving scalar wave and elastodynamic wave propagation problems using BEM. Three different BEM approaches for dynamic problems are introduced and distinguished from each other. Early research on dynamic BEM and current research for improving its accuracy, efficiency and stability are described to give the context of the research in this thesis. The contribution as well as disadvantages of these researches are summarized one by one. Then, the literature review moves to cover the field of error estimation, which introduces three main strategies: residue methods, gradient methods and re-solution methods. Consequently, this is followed by three mesh refinement strategies, h- adaptive, p-adaptive and new shape functions based on the partition of unity. A literature review of papers concerned with space-time concepts describes the current state of research in this field. Finally, the current research on parallel computing in BEM is surveyed.

In Chapter 3, fundamental mathematical theories are introduced to build a sound foundation for the later implementations. First, the governing PDE equations for scalar wave propagation and elastodynamics are presented. After introducing the reciprocal relations and Green's functions, the solutions for those PDEs, in the form of integral representation formulas, are derived for scalar wave and elastodynamics problems. By taking the limit of the integral when source points approach the boundary, boundary integral equations are obtained. These boundary integral equations are the starting point for the numerical solutions in the later chapters.

An adaptive dynamic BEM solver for 2D scalar wave problems is developed in Chapter 4. First, a new boundary integral equation is derived to consider the general variation of boundary values in space-time. Then the boundary integral equation is discretized in both space and time. The evaluation of the time integral and the space integral is described, which paves the way for the numerical solution.

Adaptivity of the BEM solver is introduced by choosing proper error indicators and adaptive schemes accordingly. Finally, numerical examples are given to demonstrate the effectiveness of the new algorithms for 2D wave propagation problems.

In Chapter 5, after introducing curved triangular elements in 3D, the BEM integral equation for 3D scalar wave problems is discretized in space-time. The significant difference between 2D and 3D problems is that only part of the boundary is integrated for Green's functions and this part is different in each time step for 3D case. This triggers a totally different BEM solution for the time integral and the space integral for 3D scalar wave problems. Elements in 3D space are indexed, like books in a library, to accelerate the spatial search to decide which part of the boundary mesh should be integrated in different time steps. Then, gradient-based and two-solution-based error indicators are used to locate moving high-gradient areas, and a triangular element refinement based on longest edge propagation path (LEPP) is employed to improve solution accuracy while retaining computational efficiency. Local time stepping is designed to fully employ space-time adaptivity. Then, numerical implementation and programming issues, such as programme structures, geometry modeling, large memory management, and fast space-search (based on sphere trees etc.) are described in detail. We apply the method to solve problems of wave propagation in a 3D bar. Compared with traditional dynamic BEM, the solution is more accurate, less artificially-damped and more stable. Finally, we compute a problem of an acoustic field inside a car to demonstrate that a reasonably accurate simulation for this complex problem can be attained using an ordinary desktop PC.

In Chapter 6, the same approach is applied to the BEM integral equation for 3D elastodynamic problems. Basic strategies are the same for the scalar wave case, but more emphasis is put on how to compute the space integral and the time integral for the vector fields, and how to compute the boundary stresses. Of course, some modifications of gradient-based & resolution-based error indicators, space refinement schemes and local time stepping for 3D elastodynamic problems are also described in detail. We apply the method to solve the problems of a 3D spherical cavity under various explosive loadings, and the simulation of a rigid foundation in a half-space under periodic loading. The accuracy and efficiency of the new method is

demonstrated in these examples.

In Chapter 7, an implementation of parallel BEM on Linux clusters is developed to further improve the efficiency of dynamic BEM. First, some useful strategies of BEM parallelization are introduced. Then, based on domain decomposition in space-time, the boundary domain is divided into several sub-domains, and can be computed in parallel to accelerate the calculation of the elements of matrices $[G]$ and $[H]$. Time steps are also divided into several *time ranges* to compute the past influence vector $\{FI\}$ in parallel to speed up the whole solution. The numerical implementation and programming issues for these parallel algorithms are described in some detail. We apply the parallel BEM solver to model a 3D underground explosion of a complex geometry. It is shown that the parallel strategy has the potential to solve large-scale dynamic BEM problems.

In the final Chapter, a statement is given of the findings and conclusions reached from the results presented in previous Chapters. A summary of the overall performance of the adaptive dynamic BEM solver is presented and comparisons are made with the uniform dynamic BEM solver in order to highlight the advantages and disadvantages of two approaches. In the final section of the thesis, we outline possible development of a time-domain elastoplastic BEM solver and a wavelet-based adaptive scheme for time-domain BEM.

Chapter 2

Literature Review

2.1 Introduction

Wave phenomena are ubiquitous in science and engineering, from applications in elastodynamics, hydrodynamics, through seismic and acoustic wave propagation, telecommunications, non-destructive testing, geological exploration, radar, sonar etc. Wave propagation and elastodynamic modeling of the interaction of waves with natural and man-made bodies is the key problem in many important engineering applications.

For example, one obvious civil engineering applications is the simulation of earthquake waves traveling through the earth. Clearly, we may expect to improve the design of safer and earthquake resistant civil structures if this basic problem can be modelled effectively. Elastodynamic equations are also used in geosciences both for geological prospecting and for assessing the effect of fault movement and fracture propagation on surface structures and mining excavations.

Another related application is the assessment of the interaction between soil and structural foundations due to dynamic imposed load or incoming seismic waves. Three-dimensional elastodynamic models are needed to capture wave propagation phenomena in the semi-infinite domain.

Further important applications are the accurate calculation of wave fields caused by reflection from complex geometry objects such as airplanes, theaters etc. These calculations are essential in forming high-resolution images, improving acoustic per-

formance etc.

A mathematical model for linear wave propagation and scattering was developed quite early. The French mathematician, *D'Alembert,* derived the "wave equation" as early as in 1747 when studying the vibration of a violin. To simulate phenomenon based on wave physics requires the solution of an appropriate wave equation such as Maxwell's equations, acoustic or elastodynamic wave equation. For all but the simplest geometry or media, this requires the use of numerical computational techniques.

Computational techniques for wave equations were developed from the 1960s. In the early days, these were numerical methods such as low order finite difference methods (FDM), or boundary integral equation (BEM) methods for time-harmonic scattering. The desire to solve more complex problems revealed weaknesses in classical wave equation solvers. Domain type numerical methods, such as FDM or FEM-based methods, require many nodes to accurately represent the field. Also, non-reflective absorbing artificial boundaries need to be created for FDM and FEM for infinite or semi-infinite domains.

Modelling impulse wave propagation introduces further numerical difficulties for domain type wave equation solvers. For these solvers, space elements must be small enough to capture the impulse (usually 8-10 nodes per wavelength are needed) and stability consideration demands that time steps must be small as well. Thus computational magnitude soon becomes so large that it is beyond the reach of the most powerful computer today. Thus, computational limitations mean that more efficient methods are needed to solve impulse wave propagation.

The boundary element method offers some advantages for wave propagation and elastodynamic problems. First, in the boundary element method, the problem scale is much smaller since only the boundaries are discretized, and the boundary condition of the open domain is naturally satisfied by the Green's function. Second, the notorious problem of numerical damping in the wave propagation problem is less significant because the method employs fewer nodes. Thirdly, the BEM provides efficient representation of cracks and geological features such as faults and parting planes, making it a powerful analysis tool for crack wave scattering prob-

lems. Finally, the boundary-only mesh in BEM offers advantages in wave scattering problems involving moving boundaries.

This chapter introduces the three most widely-used methods for dynamic BEM (Section 2.2). Then we discuss historical developments in the time-domain BEM theories (Section 2.3). We also comment on unsolved problems in each technique. There follows a literature review on error estimation and adaptive schemes in the context of BEM (Section 2.4). Latest developments of space-time concepts in numerical methods is discussed in Section 2.5. Finally, parallel computing in BEM is discussed in section 2.6.

2.2 Three dynamic boundary element methods

There are three BEM approaches to solve the wave equation and the elastodynamic equation. i.e., the Laplace transform method, the time-stepping method and the space-time integral equation.

2.2.1 Laplace transform method

The Laplace transform method transforms the time-dependent PDEs from the time domain into the frequency domain. For each fixed frequency, a boundary integral equation in the frequency domain is solved in a similar way to the static problem. Finally the solution is transformed back to the time domain by employing the inverse Laplace transform (Cruse, 1968, Cruse & Rizzo, 1968 [16]).

The advantage of this method is that dynamic responses can be obtained at a specific time without referring to all previous time steps. A disadvantage of the method is that computation of the numerical Laplace transform and its inverse transform is laborious. It would be very time-consuming if we want to calculate the time history of transient dynamic response for a longer time. Furthermore, since it is based on the Laplace transform, it is not valid for nonlinear analysis.

2.2.2 Time-stepping method

The time-stepping method is similar to the time-stepping FEM. It employs the technique of the separation of variables in space and time. The space differential operators are solved by using BEM while the time differential operator is usually discretized using finite differences [20].

There is a difficulty in the time-stepping method. Since the solution after a time step does not vanish inside the domain, it is necessary to include the volume integral in the equation in the next time step. However, the discretization of the domain destroys BEM's elegance of boundary mesh only. In order to do the justice to the name of BEM, instead of using a volume discretization, some radial basis functions are employed to transform the volume integral onto the boundary. This is usually called Dual Reciprocity method. (Nardini, Brebbia 1982 [58]).

Comment: the advantage of Dual Reciprocity BEM (DR-BEM) is that it solves dynamic problems using static Green's function. All widely-used time-stepping methods such as the Newmark method and the Runge-Kutta method can be employed. However, the smooth radial basis functions are used to represent the solution in the volume domain. Thus, they are usually inadequate to describe high frequencies wave accurately. It yields worse results for impulse problems than the standard BEM in the most cases. [3]

2.2.3 Space-time integral equations.

The time-domain BEM has some advantages over frequency domain approaches. For example, the latter is not well suited for impulse problems, and can not deal with non-linearity.

The time-domain method employs the Stokes solution as the fundamental solution, and makes use of Betti's reciprocity law to derive the boundary integral expression in space-time. Its principal disadvantage is that it is more computationally demanding because it is necessary to recalculate the matrix at each time step. The causality law implies that the integral equations at later times are influenced by events at early times. Numerical methods constructed from these space-time

boundary integral equations are global in time. That is, one must compute the solution for all preceding time steps to obtain the current solution. The space-time boundary is the spatial boundary extruded in the time dimension and therefore has one dimension more than the boundary of the spatial domain. This means a substantial increase in complexity. Thus, the system matrix is much larger and consists of high-dimensional integrals.

While the increased memory demands cannot be completely avoided, certain special features of the problem mitigate the problem:

- The system matrices are usually highly sparse due to the convolution of the Dirac delta function in time.
- For similar reasons, the domain of integration does not extend over the whole boundary of the space-time cylinder, but only over its intersection with the surface of the backward wave cone. Here the integrals are of the same dimensionality as for the static problems, and that current response is not influenced by events which took place indefinitely far into the past. These "retarded potential" integral equations are of importance for the scalar wave equation in three space dimensions and to a certain extent for equations in elastodynamics. However, it is interesting to point out that this phenomenon does not apply to the wave equation in two space dimension, nor for the heat equation.
- When low order basis functions in time are used, the system matrices are of block-triangular Toeplitz form. Matrix solution proceeds one block inverse at a time and the whole system solution remains manageable.

2.3 Dynamic BEM Research

2.3.1 Development of boundary integral theories for dynamics

The mathematical foundations of the Boundary Element Method can be traced back to a time much earlier than the invention of modern electronic computer.

From the 18th century to the early 20th century, basic ideas were formed in the development of the potential theory, Green's function theory, and integral equation theories. The solutions of some physics problems could be written either in the form of differential equations (Newtonian system) , or in the form of integral equations (Hamilton system). This reflects the fact that physical principles can be expressed as the balance of certain physical quantities, or as the minimal of certain functional. In the early 20th century, some numerical solutions using integral equations were obtained without using electronic computers, but it was after computers became popular that it began to be widely studied by researchers.

The mathematical principles behind dynamic BEM analysis are relatively simple. The dynamic response due to a point impulse load in space and time is obtained from the Green's function. According to the linear superposition principle in linear space, the weighted sum of the dynamic response of impulses distributed on the boundary are determined in such a way in which total dynamic response satisfy both the prescribed boundary conditions and the initial conditions.

2.3.2 Early dynamic BEM research

It is interesting to note that the dynamic BEM was developed simultaneously with its static counter-part. The earliest direct BEM formulation can be traced back to three papers by Jaswon (1963) [39], Symm (1963) [79] and Jaswon & Ponter (1963) [40] for static potential problems, and to a paper by Rizzo (1967) [72] for elastostatics. The earliest development of the Boundary Element Method in dynamics is attributed to Shaw (1962) [81] and Banaugh & Goldsmith (1963) [4] for their papers on acoustic and elastodynamic problems using the integral equations. It was in 1968 when Cruse & Rizzo (1968) [16] first derived the direct BEM formulation to solve transient elastodynamics problems. Cruse wrote his PhD dissertation (1968, only 40 pages long) using a to introduce a Laplace transform method to solve BEM problems in the frequency domain. Manolis & Beskos (1981) [51] extended this method to achieve higher accuracy. Niwa (1975, 1976) [60] and Dominguez (1978) [20] solved transient problems in the context of soil-structure interaction, by means of the Fourier transform and a frequency domain BEM formulation.

The time-domain method was developed much later than the frequency domain method. The first time-domain Boundary Element formulation (Cole, 1978) [15] was used to solve transient 2D elastodynamic problems for the anti-plane case. A general boundary element scheme which employs the 2D time-dependent fundamental solution to solve the 2D transient elastodynamics problem was derived by Mansur & Brebbia (1983) [54]. Later, similar results for 2D scalar wave and elastodynamic wave propagation were developed by Banerjee, Ahmad & Israil (1988) [2]. The 3D elastodynamic time-domain BEM formulation and implementation was developed by Manolis & Beskos (1981) [52] in the context of 3D dynamic soil-structure interaction problems.

Comments: These early researches lacked a systematic analysis of error estimation, convergence and stability. Instabilities were observed even for uniform meshes whenever space-time ratio β fell outside a specific range. This restricted these methods to elementary problems with the simple geometries.

2.3.3 Improving accuracies

Over the last decades, there have been increasing efforts to develop more efficient solution techniques. It has been found that higher-order methods are very attractive for decreasing the grid density needed to model wave propagation. This also has led to the development of adaptive boundary element methods (in section 2.4) and high order discontinuous Galerkin boundary element methods (in section2.5) for short wave scattering. To address these problems, there are three principal strategies.

- First, accuracy can be improved by employing higher-order spatial and temporal interpolation schemes. Various combinations of spatial and temporal interpolation schemes have been tried to improve accuracy and stability, such as combinations of constant, linear, quadratic functions and cubic B-spline interpolation etc.

- Second, some methods have been borrowed from some widely-used numerical techniques in FEM and FDM, such as the linear θ method.

- Third, some special features of the boundary integral equation have been used to derive some extra equations to improve stability.

To improve accuracy, Rizos and Karabalis (1994) [70] developed a B-spline interpolation scheme and a direct time domain BEM formulation for general 3-D elastodynamic problems. Instead of using the Dirac delta function, body forces are expressed in terms of B-spline functions in time to deduce the Green's function. Higher order spatial and temporal discretization schemes were applied to the boundary integral equations. This yields a time marching solution for the characteristic response of the system due to excitation by a B-spline time distribution, which can be extended to any transient load function.

Comment: this method can produce more accurate results, but it is much more complicated to use B-splines than the normal one.

Karabalis (1991) [43] employed a direct time domain boundary element method to solve problems in soil-structure interaction. Constant elements were used for the time marching scheme while constant, linear and quadratic elements were used in the space domain. Dominguez (1993) [20] presented a 2D formulation that included isoparametric quadratic elements with constant and linear temporal variations of displacements. Wang & Wang (1996) [86] introduced quadratic time interpolation schemes for 2D elastodynamic problems. Israil & Banerjee [38] described a 2D multi-region transient elastodynamic BEM formulation that employs isoparametric quadratic spatial elements and constant and linear temporal variation of traction and displacement respectively. Birgisson & Crouch (1998) [7] solve the similar problem using straight-line elements with a piecewise quadratic variation in space and linear variation in time. The advantage of using straight-line elements is that all integrals can be performed analytically.

Comments: These methods mainly introduced higher order interpolation in space and in time. However, most addressed 2D problems, which severely limit their usefulness.

Walker & Bluck (2002) [85] developed a time-domain boundary integral equation (BEM) solution for large electromagnetic scattering problems. It employs isoparametric curvilinear quadratic elements to model fields, geometry, and time depen-

dence. The approach is implicit, which seems to provide stability and also permits arbitrary local mesh refinement to model geometrically difficult regions.

Comments: The implicit quadratic shape functions in the time domain is claimed to offer unconditional stability. However, this conclusion is widely doubted [19].

2.3.4 Fast BEM solution methods

The boundary element method produces non-symmetric, dense matrices, which are computationally expensive to solve using direct methods. Gauss elimination requires $O(N^3)$ arithmetical operations for systems with N degrees of freedom. Even if iterative linear equation solvers are used, such as the GMRES or Lanczos method, the complexity is still of order $O(N^2)$. However, in the BEM matrices, entries are almost zero when elements are far away from the collocation points. Based on this idea, some very fast solvers of BEM of $O(N\,logN)$ have been developed in the past decade. The three most widely-used methods are: panel clustering, fast multi-pole method (FMM) and wavelet transform methods.

2.3.4.1 Panel clustering

The panel clustering method was mainly developed by a German group. Hackbusch & Nowak (1989) [33] gave an algorithm of $O(N\,(logN)^{d+2})$ with the help of an expansion of Green's functions in the far field and decompositions of the spatial domain with " panels " (d is a dimension of a problem). The panel clustering method is similar to the FMM method, which will be described in the following section.

2.3.4.2 Fast multi-pole method

The Fast multi-pole method (FMM) was first introduced by Rokhlin [32] as a fast solution method for integral equations for the two-dimensional Laplace equation. In Rokhlin's paper [32] the term *FMM* did not appear but the main framework of FMM was constructed. Later, Greengard [32] refined the algorithm, and applied FMM to two and three-dimensional N-body problems with Coulomb's potential and showed the applicability of FMM to various fields. Rokhlin uses FMM in conjunction

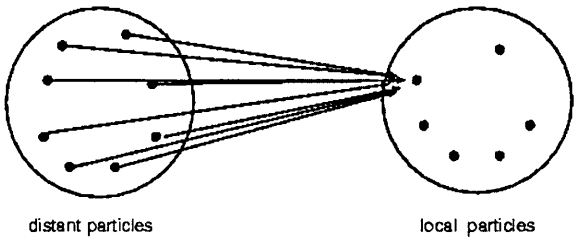


Figure 2.1: Illustration of Fast Multi-pole method (after Nishimura, Japan [57])

with an iterative solver to reduce the computational complexity for matrix-vector multiplication from $O(N^2)$ to $O(N)$. Rokhlin used multi-pole moments to represent distant collocation points in BEM and introduced a local expansion to evaluate the contribution from distant collocation points in the form of a series. The multi-pole moment associated with a distant collocation point can be translated into the coefficient of the local expansion associated with a local point (see Fig. 2.1). Greengard introduced the hierarchical decomposition of a spatial domain with a quad-tree in two dimensions and an oct-tree in three dimensions to carry out efficient and systematic grouping of collocation points with tree structures.

FMM is becoming the mainstream method to solve medium and large scale problems using BEM. It was applied to solve the 2D & 3D Laplace equation, the 2D & 3D elastostatics, and the 2D & 3D elastodynamics. Because BEM is suitable for wave analysis in the infinite domain, applications of FMM-BEM to large-scale wave problems, particularly to acoustical and electromagnetic scattering problems, have been investigated by many researchers: such as Rokhlin, Lu & Chew, Song & Chew, Fukui & Katsumoto, Yoshida [57].

2.3.4.3 Wavelet transforms

Wavelet-based methods were introduced by Beylkin (1994) [6] for the compression of large matrices in BEM. Wavelet transforms provide a fast solver which can be implemented as a black box in existing BEM codes, requiring little code change. Fast wavelet transforms remove the redundancy of far-field information in BEM by compressing these data after the matrices have been assembled by standard means.

Although less efficient than the other two techniques mentioned before, it is easier to implement.

2.3.5 Stability of dynamic BEM

The conventional time-domain BEM exhibits numerical instabilities at later times, particularly when more complicated geometries and loading configurations are modeled. For example, the time domain direct BEM formulations proposed by Mansur for 2D problems [12] exhibit increased oscillation at later times, indicating potential instability. Dominguez [20] put his 2D time-domain elastodynamic BEM code in public domain. When this code is used to solve a benchmark problem of a circle embedded in the infinite domain with a Heaviside load, it yields unstable results after 2000 time steps, see Fig. 2.2 [26]. Manolis [26] presented a 3D time domain direct BEM, with results for less than 25 time steps. This scheme used repeated averaging in time in order to filter out the oscillations in the results.

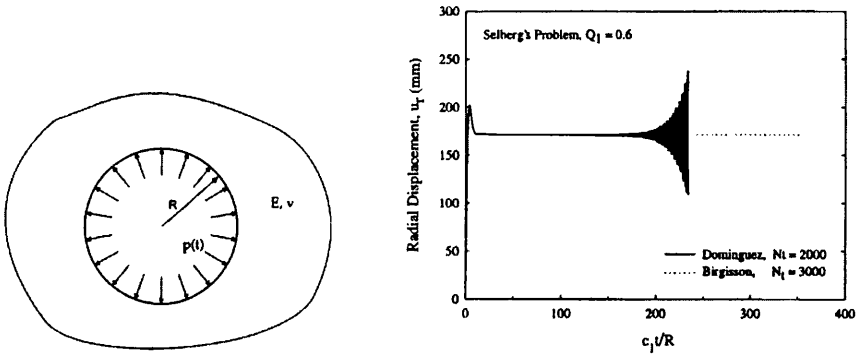
Some stable time-stepping strategies were borrowed from FEM to improve the stability of transient dynamic BEM solution, such as the linear-θ method by Yu (1998) [90]. The procedure is similar to that of the Wilson-θ method, but whereas the Wilson-θ method assumes linear time variation of acceleration, his method assumes linear time variation for both displacements and tractions for elastodynamic waves. The best values of θ for stable results vary at different time steps.

Mansur & Carrer (1998) [53] developed the time-discontinuous traction method. The additional unknown variables introduced by discontinuities can be obtained by coupling the standard boundary integral equation for displacement and the one for traction.

Comment: It is computationally expensive to invoke the hypersingular boundary integral equation for traction.

Marrero & Dominguez (2002) [55] combined boundary integral formulations for several time steps, assuming that velocities are constant (in time).

For stability in dynamic FEM and FDM, the time step must be smaller than a certain critical value. Peirce & Siebrits (1996,1999) [64] were the first researchers to investigate the stability problem of elastodynamics BEM. A sequence of simpler



(a) (b)

Figure 2.2: a) Circular cavity suddenly loaded with normal traction b) Unstable results after 2000 time steps (A. Frangi and G. Novati, [26])

time-dependent PDE problems, which is similar to the dynamic BEM equations, were analyzed by using the z-transform method which reduces the time convolution equations to simple algebraic equations. Their analysis demonstrated that the source of the exponential instability is that the Green's function does not decay rapidly enough with space and time. Therefore, an over-estimated coupling between neighboring elements and remote ones leads to as situation in which the response of neighboring elements to a stimulus from a given element is more energetic than the original signal. This positive feedback causes the exponential growth observed in numerical solutions. Their analysis showed that eliminating this problem needs enhanced self effects (enlarge the diagonal terms in BEM matrices) to achieve more stable schemes.

Another cause of instability is that continuously moving wave fronts are poorly approximated by discrete time steps. High gradient areas such as wave fronts are unknown *a priori,* and can not be properly approximated by ordinary shape functions in uniform meshes. Spurious oscillations or even instabilities are often observed in these regions. Alternatively, dispersion occurs if lower-order approximations or bigger time steps are used to damp these oscillations. These high gradient areas must be tracked and well represented by adaptive schemes. This is the reason why adaptive schemes in space-time have been chosen to improve the stability of dynamic BEM in this research project.

2.4 Error Estimation & Adaptive Schemes

2.4.1 Objectives of error estimations and adaptive schemes

In the dynamic BEM, uniform meshes can produce reasonable results if solution functions are smooth and regular. However, they will be inadequate if local high-gradients appear in the solution. Adaptive schemes are needed to construct an optimum mesh for domains where high resolution is needed. Adaptive schemes for BEM involve:

- Mathematical models: Mathematical models and formulations are derived. Geometries and boundary conditions are defined for the problem domain.
- Mesh initialization: Create the initial mesh for the boundary element analysis.
- Boundary element analysis: Boundary element analysis is performed for the current mesh. The solution for displacements or tractions are obtained on the boundary.
- Error estimation & Convergence satisfaction: The errors of the boundary element solutions are estimated at each step. The convergence criterion is checked to decide whether further refinement is needed.
- Adaptive schemes and mesh refinement: The element errors computed above are used to determine which elements are to be refined according to the prescribed criterion. Then those elements are refined according to h- or p- refinement schemes.

Accurate error estimation plays a key role in the efficiency of the algorithm. Since the 1990s, a substantial of literature have been devoted to adaptive methods for BEM. Most of them are devoted to the formulation and implementation of better error estimations. Some error estimations which are the most relevant to the scalar wave and elastodynamics are described Here. Some observations and heuristic formulas are presented since it is far from a mature theory and is subject to ongoing research.

2.4.2 Error estimation in space

2.4.2.1 Residue methods

Consider the 2D potential problem:

$$\nabla^2 u = 0 \quad on\, \Omega \tag{2.1}$$

$$u = \bar{u} \quad on\, \Gamma_u \qquad q = \frac{\partial u}{\partial n} = \bar{q} \quad on\, \Gamma_q \tag{2.2}$$

where u and q are the potential and its flux respectively. Taking u^* as the fundamental solution, then the integral representation of the boundary value problem is:

$$c_i u_i - \int_\Gamma [qu^* - uq^*] d\Gamma = 0 \tag{2.3}$$

Discretizing the boundary Γ , we obtain:

$$L(u) = c_i u_i - \sum_j^N \int_{\Gamma_j} [qu^* - uq^*] d\Gamma = 0 \tag{2.4}$$

where $\Gamma_j$is a piece of boundary, N is the total number of element. Assuming the approximate solution of u and q is $\hat{u}$ and $\hat{q}$, the potential and flux error, as shown in Fig. 2.3, the residual error of displacements e_u are:

$$e_u = L(\hat{u}) = c_i \hat{u}_i - \sum_j^N \int_{\Gamma_j} [\hat{q}u^* - \hat{u}q^*] d\Gamma = 0 \tag{2.5}$$

If a Galerkin BEM is employed, it can be mathematically proved that the solution errors are bounded by the residue of the boundary integral equation:

$$c_1 \| R \| \leq \| e_u \| \leq c_2 \| R \| \tag{2.6}$$

where $c_1$and $c_2$are two contants. Thus, the residual can be used as an efficient error estimator. A global error estimate can be defined as some norm of point-wise errors. To choose a proper norm is subtle since it varies from one problem to another.

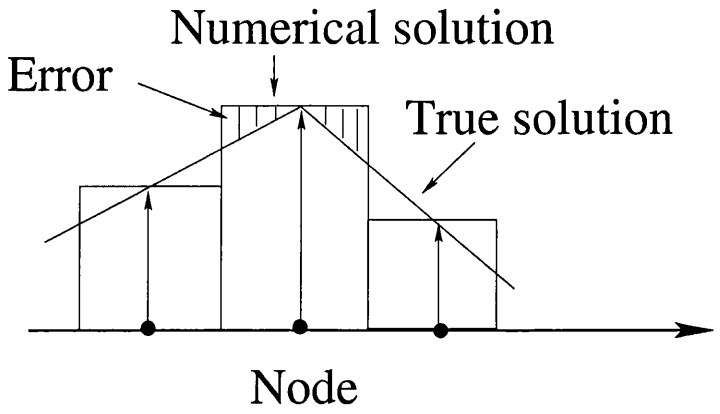


Figure 2.3: Illustration of residue-based error estimation

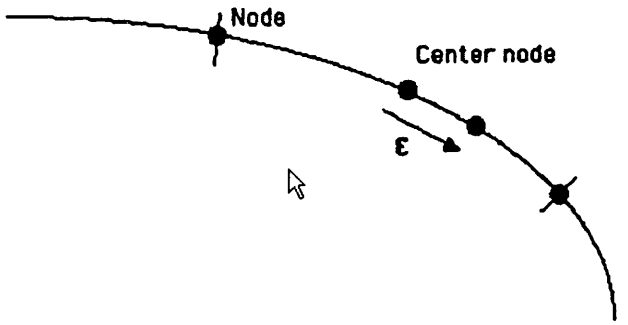


Figure 2.4: Error estimation based on sensitivity

According the Hiao (2004) [35], a H^0 Norm is better for smooth functions while a $H^{\frac{1}{2}}$ norm is better for non-smooth ones.

Such residue-based error estimates are widely-used in FEM. However, if the Galerkin BEM is used, a fifth-order integral is needed, which is too expensive for most practical problems. This is the reason why BEM is usually formulated as a collocation method. However, in the collocation BEM, residue errors disappear at collocation points as shown by Eq. 2.5. To overcome this problem, Guiggiani & Paulino [63] developed error indicators based on nodal sensitivity. The initial analysis is done by the ordinary boundary elements. Then nodes are displaced slightly to get another solution. (e.g. for constant elements, nodes would be displaced off-center a little, as shown in Fig. 2.4). The difference between the two solutions is the residue of this point. But no rigorous mathematical formula like Eq. 2.6 can be proved. It can only be demonstrated numerically.

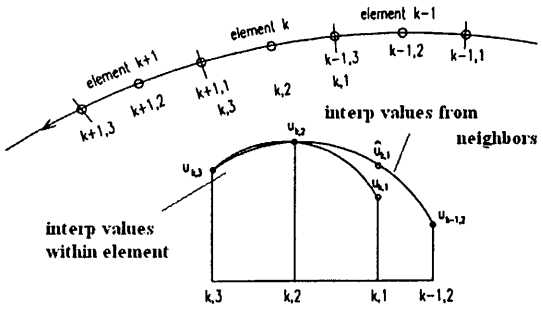


Figure 2.5: Error indicators based on quadratic interpolation

Super-convergence based error estimation

Zienkiewicz (1992) [61] proposed an error estimation for FEM based on super-convergence of the recovered stress of the FEM solution. For integral equations, the residual can be used as an error estimator provided that the BEM is super-convergent (Geng, 2001) [30]. When the Galerkin BEM is used, the super-convergence of BEM can be established easily under certain conditions. However, this is more difficult for collocation BEM. Geng also showed that it is important to consider numerical integration errors, which can destroy super-convergence and render the whole method useless.

Comments: Because some strong conditions must be satisfied, its application to practical problems is limited.

2.4.2.2 Interpolation and Gradient methods

Zhao (1996) [91] developed a simple error indicator based on quadratic interpolation. The method assumes that in the vicinity of a quadratic element, the state variable should vary quadratically, too. An error indicator can be derived by calculating the difference between the BEM solution and the interpolated solution. The method does not require extensive computation and is demonstrated by Fig. 2.5.

Comments: This method is very difficult to extend to the 3D situation when interpolation is much more complex.

Rencis & Kita [23] developed an interpolation error estimation scheme. Assuming that a higher-order interpolation yields a more accurate solution than a lower-order

one, the difference between the two solutions is considered as the error. This method is widely used since it is simple. However, it is uncertain that a higher-order solution is necessarily more accurate than a lower-order solution.

Gradient error type

Errors usually arise because lower-order methods can not approximate the higher-order terms in the true solution. A gradient-based error estimation method simply assumes that errors arise near high gradient areas. The error indicator is based on the local magnitude of the first or second derivatives of displacements or tractions in the solution.

$$e_i^u = u_i - u_i^h = \triangle u_i \qquad e_i^p = p_i - p_i^h = \triangle p_i$$

$$\|e_i^u\| = \left[\frac{\int \left(\Sigma \frac{\partial N}{\partial x} \triangle u_i\right)^2 + \left(\Sigma \frac{\partial N}{\partial y} \triangle u_i\right)^2 d\Gamma_i}{\int d\Gamma_i} \right]^{\frac{1}{2}} \tag{2.7}$$

where $e_i^u$and e_i^p are the point-wise errors of displacements and tractions; u_i and p_i are point-wise solutions of displacements and tractions; u_i and p_i are point-wise approximated solutions of displacements and tractions on the discrete mesh; N are shape functions; Γ_i is the ith boundary element; $\|\|$ is the L_2 norm.

Comments: This method is simple to implement and is used widely in error detecting on shock waves problems, etc. or strong impulse waves. However, it is not particularly accurate for coarse meshes.

2.4.2.3 Re-solution methods

Some researchers [45] have suggested schemes in which errors are calculated based on re-solution. An ordinary analysis is carried out first and then a different scheme is used in the re-solution. e.g., from N elements to $2N$ elements, or the same mesh from lower-order shape functions to higher-order ones, or from ordinary Lagrangian shape functions to derivative-continuous Hermite shape functions, etc.

Comments: This method is simple to implement and is used widely. The disad-

vantage is that the computational cost is high.

2.4.3 Error estimation in time

Time-step adaptivity has been used for ordinary differential equations for many years. Dynamic FEM can be simplified to coupled or uncoupled ODE problems in time, which can be solved by recursive time-stepping methods, explicitly or implicitly. The solution for the next time step is determined from the solutions of a few previous time steps. This is not the case for dynamic BEM. Since Green's functions are global operators, all time steps are coupled. Although we do have error estimates for static Galerkin BEM and first-order time-dependent BEM such as heat conduction, it is quite difficult to develop an error estimate for second order time-dependent BEM, such as the wave equation. Thus, instead of rigorous theory, heuristic methods will be used to calculate error estimates in time.

2.4.3.1 Post processing based error indicators

This method is the counterpart in time for Zienkiewicz's error estimate in space. After we obtain the solution for each time step, we can post-process the solution to get improved estimates which we shall consider to be more accurate. Wiberg (1993, 1994) [87] postprocessed the displacement vectors u, stress tensor σ and velocity vector $\frac{du}{dt}$ based on the assumption that the third-order time derivatives of displacements t vary linearly over each time step. As a posterior temporal error estimate of displacement and velocity is the difference between the original values u_h, σ_h, $\frac{du_h}{dt}$ and the postprocessed ones u, σ, $\frac{du}{dt}$. The temporal error in the total energy norm can be written as:

$$\|e_t\| \simeq \|u - u_h\| = \left[\int_\Omega (\frac{du}{dt} - \frac{du_h}{dt})\rho(\frac{du}{dt} - \frac{du_h}{dt})d\Omega + \int_\Omega (\sigma - \sigma_h)D^{-1}(\sigma - \sigma_h)d\Omega\right]^{1/2} \tag{2.8}$$

where ρ is the density of the material and D is the constitutive matrix.

Comments: This method is efficient and easy to implement. However, the as-

sumption of linear third-order time derivatives of the displacements is not universally true for most dynamic problems.

2.4.3.2 Re-solution based error indicators

In the re-solution method, the time-steps are chosen such that the local error in each step satisfies the prescribed error tolerance. The local error is estimated by comparing two different time discretizations. If the estimated error is too large, then the time step is decreased. If it is smaller than the tolerance, then the step size can be increased. In this way, the balance between the accuracy and the efficiency is maintained. Also, no initial guess of the step has to be supplied.

Another strategy of re-solution is to employ different order of interpolations. The solution based on the higher-order interpolation is assumed to be more accurate than that of lower-order one. An error indicator can be calculated as the difference between the two solutions.

Comments: This simple method is widely used, including the research of the author.

2.4.3.3 Discontinuous Galerkin BEM error indicators

In the discontinuous Galerkin BEM [14], a natural choice of the temporal error indicator is the local temporal jump term across the interfaces between space-time sub-domains, as shown in Fig. 2.6.

Comments: This is an efficient error estimator which has been mathematically shown to converge to the real solution error. However, the method is confined to the discontinuous Galerkin method only.

2.4.4 Adaptive schemes for static problems

After calculating error estimates for meshes using various methods, we will move on to show how to refine those elements with large errors. Suppose that u is the true solution to an n-dimensional elliptic variational problem of order m; also assume that u^h is the numerical solution in the discrete space S^h and k is the order of the

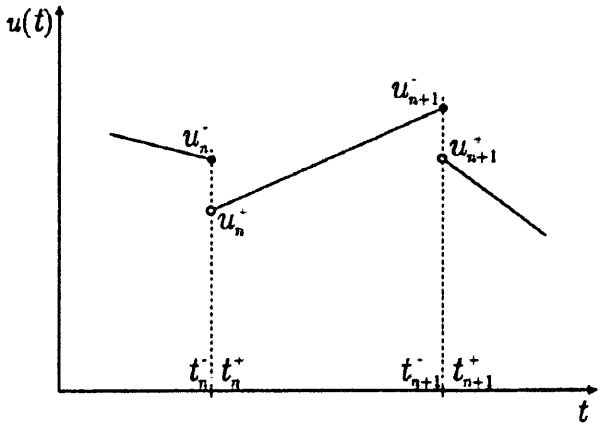


Figure 2.6: Illustration of time-discontinuous approximation

polynomial in shape functions. An *a priori* error estimation formula in the bilinear form is [24]:

$$\alpha(u - u^h,\, u - u^h) \leq C^2 h^{2(k-m)} \mid u \mid^2_{,k} \tag{2.9}$$

where C is a constant, $\mid u \mid_{,k}$ is the L_2 norm of kth derivative of u, $h^{2(k-m)}$ is the rate of convergence in strain energy. From this equation, it is clear that errors can be reduced either by increasing the number of the elements, or decreasing the size of the elements h, or employing higher-order shape functions.

2.4.4.1 h- adaptive and hierarchical h- adaptive schemes

The h- refinement scheme simply involves increasing the number of the elements while keeping the order of the interpolation function intact. The naive h- adaptive scheme is simple to implement, but it is very computationally expensive since the global influence matrices $[G]$, $[H]$ have to be re-computed. To solve this problem, the h- hierarchical refinement scheme, proposed by Kita, Kimaya & Parreira [46], can be implemented. Standard and h-hierarchical linear shape functions are shown in Fig. 2.7.

In the hierarchical h- refinement scheme, the initial analysis is carried out using standard shape functions. Then the h- hierarchical interpolation functions are added to the solution. We suppose

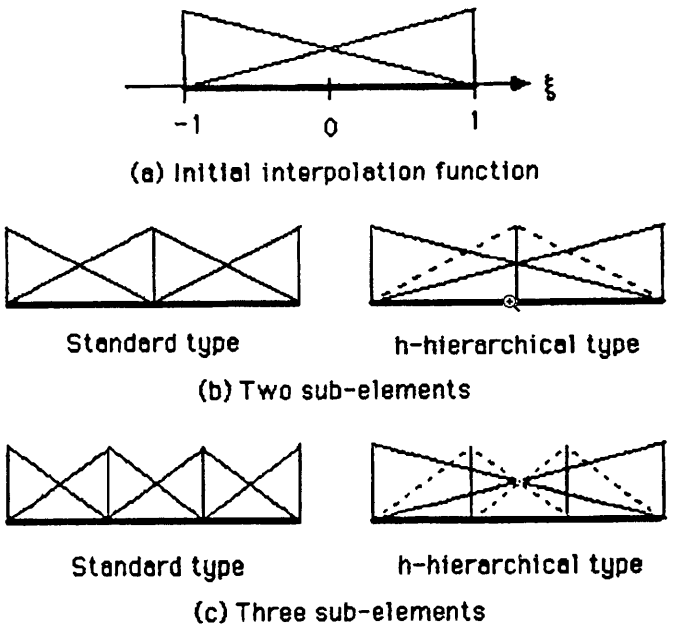


Figure 2.7: Standard and h-hierarchical linear interpolation functions

$$u = \sum N_k u_k + \sum N_\alpha u_\alpha \tag{2.10}$$

where u_α are the newly generated degrees of freedom. The matrix equation for the initial mesh is

$$A_{11} x_1 = b \tag{2.11}$$

After the refinement, we obtain the new matrix equation

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \end{Bmatrix} \tag{2.12}$$

where the coefficient matrix A_{11} remains the same and only the entries in the influence matrix for the newly-added terms (involving A_{12}, A_{21}, A_{22}) need be computed. This saves a great deal of computing time.

2.4.4.2 p- adaptive schemes

In the p-refinement scheme, the initial mesh is not refined but the order of interpolation functions are increased, everywhere or only in some elements. There are

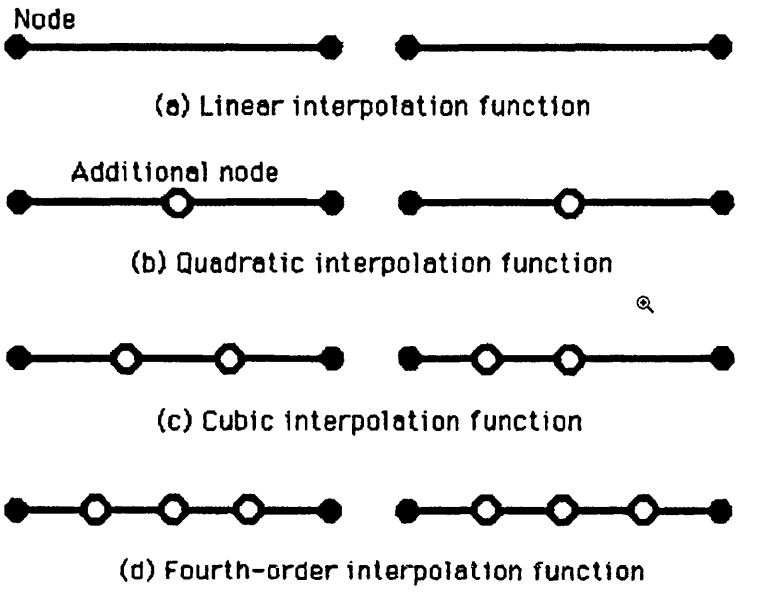


Figure 2.8: Adding new collocation points in p- hierarchical scheme

two categories of p-refinement schemes: '*the ordinary type* ' using ordinary trial functions and ' *the hierarchical type* ' using p- hierarchical trial functions.

In the p- hierarchical refinement scheme, the conforming linear trial function is used first; then higher order p- hierarchical trial functions are added in, such as the Legendre polynomials and Peano family.

Legendre polynomials $N_k = \frac{1}{2^{k-2}(k-1)!}\frac{d^{k-2}}{d\xi^{k-2}}\left[(1-\xi)^{k-1}\right]$

Peano family $N_k = \frac{1}{k!}\left(\xi^k - b\right)$

where ξ is the local coordinates, d and b are Legendre constants and Peano contants respectively.

The Peano family is more convenient to use and performs better.

When the order of the interpolation is increased, new collocation points are located on the element to be refined. There are two ways of adding new collocation points: one is ' *the symmetric type* ' and another is ' *the true hierarchical type* '. The differences between them are shown in Fig. 2.8.

2.4.4.3 hp- adaptive schemes

It is natural to combine the h- adaptive and p- adaptive schemes. Heuer, Mellado and Stephan (2005) [59] developed a hp-adaptive two-level method for Galerkin

boundary integral equations on the curved boundaries. An *a posterior* error estimate was derived from a comparison between the original solution and the half-size element mesh. A hp-adaptive algorithm is used to increase polynomial interpolation order by one and reduce element size by half. If the solution of the problem is smooth and regular, this method converges exponentially to the true solution.

Comments: The refinement scheme is very straight forward. However, it may give poor results for problems with sharp corners and other singularities.

2.4.4.4 New shape functions

Perrey-Debain and Trevelyan (2004) [65] developed wave boundary elements to deal with high frequency wave problems. Besides ordinary polynomials, they incorporated the plane wave as a shape function and obtained good results. Only 2.5 wave boundary nodes are required per wavelength, instead of the 8-10 nodes required by conventional interpolating methods.

Comment: The method offers higher accuracy using a coarse mesh. However, the plane wave shape functions are no longer orthogonal to each other and this leads to ill-conditioned system equations.

2.4.4.5 Summary of adaptive schemes

If the problem solution is smooth, p- adaptive schemes are more efficient than h-adaptive schemes. However, if non-smooth functions or singularities appear, the h-adaptive schemes are more stable, but may require an exponential increase in element numbers. The hp- adaptive scheme may be the best choice for smooth and regular solution because of its fast convergence. It is better to approximate singularities by adding singular shape functions to the element or by using graded meshes.

2.4.5 Adaptive schemes for dynamic problems

The primary objective of an adaptive scheme is to use the largest possible time step while maintaining accuracy and stability.

2.4.5.1 Moving mesh and dynamic mesh refinement

Many adaptive methods in FEM and BEM have been proposed to address general problems of this nature. Two classes of adaptive methods for dynamic problems can be distinguished; namely, (i) moving mesh methods and (ii) dynamic mesh refinement with h- , p- or hp- adaptive schemes.

In moving mesh methods [1], nodes evolve in the space-time domain and discretization of the governing equations is coupled with the moving mesh. The advantage of this method is that it uses fewer nodes and larger time steps if the nodes travel smoothly. But consequent mesh distortions can introduce considerable difficulties. Above all, controlling the movement of mesh itself is a computationally intensive task.

In dynamic mesh refinement methods the location of the original mesh is fixed. The h- adaptive method involves adding more nodes when they are necessary and removing them when they are no longer needed. The p- adaptive method captures rapidly changing functions by refining the shape functions. For example, in order to accommodate discontinuities, the Discontinuous Galerkin method was developed. However, it doesn't solve the problem of rapidly moving high gradients in general. Farhat et. al. (2001) [25] suggest using the discontinuous enrichment method to address the multi-scale problem. Chessa & Belytschko (2004) [13] proposed a space-time element of X-FEM to capture arbitrary discontinuities in the time domain. Yue & Robbins [94] observed that adaptive schemes should be applied to both space and time domains.

Comments: Moving mesh methods are popular for solving dynamic FEM, but they are not very suitable for solving dynamic BEM. Since BEM is a global operator applied on the boundaries of the problem domain, it is of little use to move nodes around to achieve better accuracies for dynamic problems. Thus, dynamic mesh refinement is used in this research.

2.4.5.2 Local time steps

If dynamic problems are solved using space adaptive schemes, it is necessary to adapt the time discretization to the space mesh as well. The time step $\triangle t$ must

be adapted to the smallest mesh size in the whole spatial domain, even if the finer elements are only concentrated in a small domain. If the spatial meshes are refined locally, the time step $\triangle t$ must also be refined to maintain the accuracy and stability of the numerical scheme. This means that even if the spatial refinement scheme is applied in a small domain, the complexity of the whole problem increases quite substantially. To avoid this problem, some researchers introduced local time steps that are adapted to the local mesh spatial refinement.

Bacry & Mallat (1992) [21]suggested a wavelet based space-time adaptive numerical method for PDEs. They focused on problems in which high resolutions are needed only in region where singularities and high gradients occur. It is necessary to adapt the time steps according to the spatial adaptive schemes in order to maintain the stability and precision of the scheme.

Piperno (2006) [67] suggested a local time-stepping discontinuous Galerkin Time Domain method for wave propagation problems. Some unstructured, locally-refined meshes were used to handle complex geometries. Usually the time step is restricted by the smallest elements in the mesh for the sake of the stability. A local-time stepping algorithm is constructed to use large time steps for most parts of the the mesh while smaller time steps are used where the elements are finer. *Comments: Local time steps are necessary to improve the overall efficiency of the adaptive scheme for dynamic BEM.*

2.5 The space-time concept in numerical methods

2.5.1 Introduction

The numerical solution of the wave problems usually leads to the discretization in space using FEM or BEM, and discretization in time using finite difference method (FDM). It is conditionally stable and therefore stability demands a small time step dictated by the smallest elements in the mesh. It causes no difficulty for a uniform mesh with a fixed time step which is prescribed according to the mesh size. However, it is impossible for adaptive schemes to keep a single time step size while refining meshes in space.

Furthermore, traditional time stepping methods will have numerical difficulties, such as severe numerical damping or instability. For example, when standard second order time stepping methods, such as Crank-Nicolson and Newmark methods, are applied for problems involving the propagation of wave impulses over a large distance and time, they exhibit significant dispersion errors as well as spurious oscillations near to the moving wave fronts [67] . Thus, space-time adaptivity is needed for the reliable and efficient numerical solution for time-dependent wave propagation.

2.5.2 Space-time research in FEM

In the field of FEM research, one of the most promising high-order space-time methods is the time-discontinuous Galerkin space-time finite element methods (DGFEM) which employs finite element discretization of the time domain as well as the usual discretization of the spatial domain (Hughes, 1988, [37]). The DGFEM possesses high-order accuracy and is unconditionally stable, and its principal features are:

- Higher-order approximations in both space and time,
- Unstructured meshes in both space and time are possible,
- Physically based dissipative mechanisms,
- Algebraic form provides natural setting for predictor/corrector and iterative solvers for high-performance parallel computation.

These features offer the promise of significant advances in efficiency, reliability, and flexibility in numerical simulation designed for time-dependent wave-propagation over large distances and time and for complex geometries. High-order space-time finite element methods are capable of delivering very high accuracies for wave propagation simulations over large distances and time, particularly for problems involving sharp gradients in the solution which typically arise in the vicinity of fluid-structure interfaces and near inhomogeneities such as structural joints, and material discontinuities. For such problems, solutions obtained with standard numerical methods have difficulty resolving the discontinuities or high gradients.

The notion of allowing discontinuous finite element approximations in time originated in the classical work of Lesaint & Raviart (1974) [62], over nearly two decades ago for first-order hyperbolic equations. Johnson and co-workers [42] generalized the theory of discontinuous finite element methods by introducing mesh-dependent norms and were able to derive a priori error estimates for first-order hyperbolic systems and conservation laws. Time-discontinuous methods have since been applied to parabolic systems and advection-diffusion systems [76] [78]. Space-time finite element methods have gained popularity for computational fluid dynamics (CFD) simulations where accurate resolution of shocks, multiscale phenomena, and moving boundaries is important [10]. The idea of using time-discontinuous space-time finite elements has been extended to second-order hyperbolic equations governing time-dependent wave propagation and in particular to the structural acoustics problems, including problems with infinite domains [5] [66] . These methods give a consistent and unified methodology for obtaining accurate and stable solutions in both time and space dimensions, and provide a solid mathematical framework for adaptive space-time solutions.

Discontinuous Galerkin methods use finite element discretizations in space and time simultaneously with basis function which are continuous in space and discontinuous in time. Discontinuous Galerkin methods are implemented over space-time slabs $S_n = \Omega \times T_n$, where Ω is the space domain and T_n are time steps. The basis functions are discontinuous across the solution at time t (Fig. 2.9). It has been shown [37] that the space-time finite element method possesses properties missing in traditional semi-discrete approaches. In particular, Discontinuous Galerkin methods often lead to A-stable, higher-order implicit time-stepping schemes and are suitable for the various type of adaptive schemes.

Wiberg, Zeng & Li [87] employed the space-time concept for error estimation and adaptivity in elastodynamics . The adaptivity in space was done by mesh refinement (h-version); by increasing the order of the approximation polynomials (p-version) or a combination of the two (hp-version). In the time domain, the integration is either made by mode superposition of the characteristic functions or by direct integration. The adaptation in time may either involve changing the global time-step (h-version)

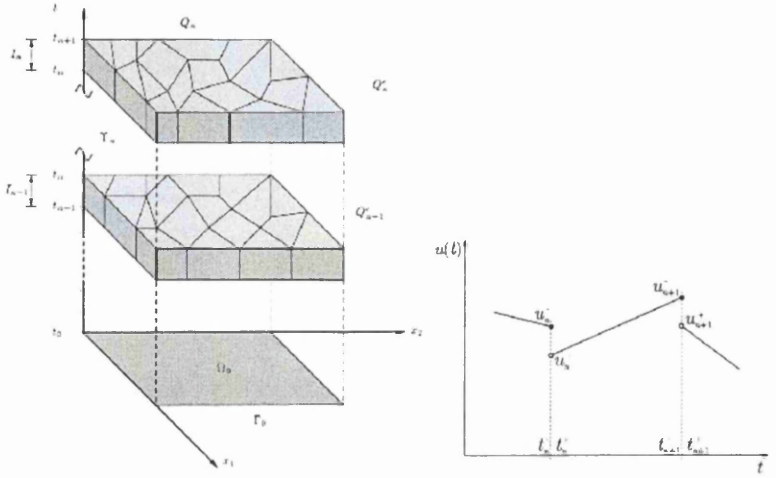


Figure 2.9: (a) Illustration of space-time adaptivity (b) Illustration of time-discontinuous approximation

or increasing the order of the approximation polynomials (p-version). They also presented some new error estimators and error indicators based on interpolation theory and total energy obtained from postprocessed stresses.

Ericksony & Guoy (1992) [87] presented an algorithm to construct space-time meshes over arbitrary spatial domains which are suitable for space-time discontinuous Galerkin FEM methods. Given an arbitrary simply meshed space domain Ω, the algorithm extruded it into the space-time domain $S_n = \Omega \times T_n$, keeping the size of the space-time elements in proportion to that of elements in space mesh. Bécache (2004) [22] also employed space-time mesh refinement for some problems in elastodynamics. He used different time step sizes for the refined spatial meshes, in order to keep the space-time ratio constant and to decrease the numerical dispersion. Yue & Robbins (2005) [94] employed an adaptive space-time scheme of finite element meshes to solve elastodynamic problems. Huang & Costanzo (2002) [34] also used space-time finite elements for the solution of elastodynamic problems with strain discontinuities.

Comments: Although the discontinuous Galerkin method offers a good framework to solve the elastodynamic problems adaptively, it is not ideally suited for modeling wave propagation in open fields. Some special infinite elements or wave-absorbing elements have to be employed to model the infinite boundary condition. Moreover, the discontinuous Galerkin FEM still does not avoid spurious oscillations around

moving wave fronts totally. Commonly, a least-square method is used to suppress these oscillations.

2.5.3 Space-time research in BEM

There is far less literature on space-time adaptive schemes in the dynamic BEM. The obstacle is obvious: coupling current time step with all previous time steps in dynamic BEM makes it far more complicated than its counterpart in FEM. Ha-Duong & Ludwig (2005) [82] present a space-time Galerkin BEM for transient acoustic scattering by an absorbing obstacle. Space-time boundary elements are implemented in the model to increase the stability for a wide range of situations.

Comment: Galerkin BEM in space-time appears to be quite cumbersome since it involves a 5th-dimensional integral (four dimensions for the space domain, one dimension for the time domain). So far, no work has been reported on the space-time collocation BEM method, which is explored in this thesis.

2.6 Parallel Computing in BEM

2.6.1 Introduction

For dynamic BEM, the computation is demanding because new influence matrices are computed at each time step. In order to obtain more efficient solutions, some acceleration technology is needed. Parallel computing techniques are the right solution to address this challenge.

However, parallel computing is a complicated process where the efficiency and accuracy of results depend on the chosen hardware and software. Efficient implementations vary from one parallel system to another. The key to achieve increased speed for numerical models is the implementation of appropriate parallel algorithms which take advantage of the specific computational features of parallel computers. Most parallel computers belong to two categories: shared memory and distributed memory computers. Shared memory systems have relatively few but very powerful CPUs which each has a large memory. Distributed memory computers consist of a

large number of ordinary computers (such as PCs) which have their own memory. The computing task will be divided and assigned to different computers, controlled by the communication between the processors. The author has a 8-CPU Linux PC cluster available to him: this is a distributed memory MIMD (multiple instruction, multiple data) parallel computer.

2.6.2 Theory of BEM parallel computing

The factors which need to be considered for optimum performance include domain decomposition, load balancing and parallel solvers. Some parameters are also introduced to evaluate the performance of parallel algorithm.

2.6.2.1 Domain decomposition

Domain decomposition means division of the original, complex domain into smaller, simpler sub-domains. The simplified solution in each sub-domain is solved by different computers. The whole solution is then reconstructed by setting up equations for the interfaces between different sub-domains. A well-designed decomposition scheme will minimize both the time spent on local computers and on inter-computer communications. Further, time spent on inter-computer communication will be lower by minimizing the boundary size of sub-domains and the total number of messages sent.

2.6.2.2 Load balancing

Load balancing is a key issue controlling the efficiency of distributed computers, such as a Linux cluster. It is more than dividing up the computing task and assigning them into each computer evenly. For example, for the current hardware implementation, communication between the computers is relatively slow. Therefore, it is necessary to optimize both the load imbalance and communication rather than simply using more CPUs.

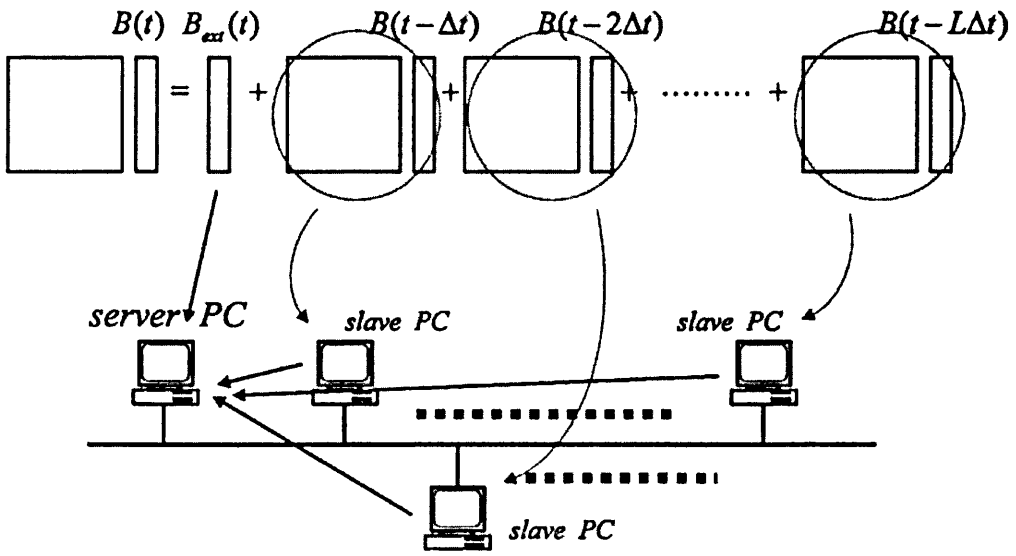


Figure 2.10: Configuration of parallel computer systems

2.6.2.3 Parallel Solvers

Besides domain decomposition, the whole BEM analysis process, such as the assembly of influence matrices, the solution of the linear system etc., can be distributed to different computers (Fig. 2.10). The linear system solver is usually the most time consuming routine in BEM codes. We choose certain highly efficient machine-specific linear system solvers for non-symmetric and dense matrices. For example, LAPACK (Linear Algebra Package) [48] is a high performance parallel library especially written for solving linear algebra problems on various parallel computers. The ScaLAPACK (Scaled LAPACK) [75] package is a subset of LAPACK routines redesigned for distributed memory MIMD parallel computers. ScaLAPACK is written in FORTRAN/C and can be freely downloaded from the Internet. Thus, using standard and sophisticated libraries as linear solvers, parallel computing performance can be substantially improved.

2.6.2.4 Performance of parallel scheme

The performance of a parallel scheme can be measured in terms of two indexes, i.e. *speed-up* and *efficiency*. The *speed-up* index is a measurement of the improvement gained by the parallel program with regard to a single processor. The *speed-up* index

is defined as

$$Speedup = \frac{user\ time\ for\ one\ processor}{user\ time\ for\ N\ processors} \tag{2.13}$$

It is controlled by both the inherent parallelism of the application and the efficiency of the system facilities. High speed-up implies that the computational time taken will be very much reduced but it may be achieved by using a large number of processors, and therefore, the efficiency may not be high. The efficiency of the scheme can be measured as speed-up per processor, that is

$$Efficiency = \frac{speedup}{number\ of\ processors\ used} \tag{2.14}$$

2.6.3 Research in BEM parallel computing

The first parallel implementation of BEM was done by Simkin (1976) [77] for an electromagnetic field problem. He solved the Maxwell equation with Dirichlet boundary condition in a circle using a shared memory computer. In early work, the parallelization of BEM mainly focused on solving the linear equation system with parallel solvers. Calitz & Du Toit (1990) [11] employed a shared memory computer to accelerate the solution of an axisymmetric electromagnetic problem. Later, parallel computing on distributed computers were introduced by Davies (1997) [18] in the field of potential problems.

Cunha & Telles (2004) [50] developed a parallel version of a code presented by Brebbia & Dominguez (1989) [8] for solving potential problems by using high performance parallel libraries such as LAPACK. In the paper, it is mentioned that the use of LAPACK lead to a reduction of 46% in the execution time.

Kreienmeyer & Stein (1997) [47] did some research on the parallel implementation of several solvers (GMRES and Bi-CGSTAB) for BEM on a MIMD parallel computer. In their paper, they concluded that for solving large non-symmetric and dense systems, iterative solvers have a superior performance compared with direct solvers such as Gauss elimination.

Inevitably, time-domain BEM requires both large memory and incurs high com-

putational cost. Kawaguchi (2003) [44] constructed a parallel computer system in which influence matrices from each past time step was stored and computed in individual PCs, and assembled on a master PC later. It achieved high speed-up because only vector values were communicated among computers.

Iterative methods are commonly used for solving large-scale systems of linear equations. González, Pena & Cabaleiro (2004) [31] employed parallel sparse approximate preconditioners to speed up the solution process.

Nishimura (2004) [80] developed a fast BEM parallel program for large scale elastodynamic problems in the time domain. A Plane Wave Time Domain Algorithm (PWTD) was used to reduce the amount of computation, which utilizes the plane wave expansion of the fundamental solution and the hierarchical structure of the space-time in the elastodynamics equation in 3D. A parallel algorithm for shared memory computers using MPI-OpenMP hybrid parallelizations was developed to enable an analysis of large-scale problems with more than 1 million spatial degrees of freedom.

2.7 Summary

In this chapter, the literature has been reviewed in the relevant areas of elastodynamic wave propagation dealing with accuracy, efficiency and stability. Since domain type numerical methods such as FEM and FDM need many elements to solve wave propagation problems in 3D open fields, and non-reflective artificial boundaries are also needed to satisfy infinite far field boundary conditions, dynamic BEM is a good choice to solve this problem. However, the conventional BEM is still very expensive since impulse waves just occupy a small area in space-time. An adaptive scheme is necessary to capture the features of moving wave fronts and to offer a higher resolution there. Not only do the meshes need to be refined in space, the time steps near the spatial refinements are also changed adaptively in order to maintain the stability and accuracy of the whole solution. Finally, literature on parallel algorithms have been reviewed in the context of improving the efficiency of dynamic BEM solvers.

Chapter 3

Boundary integral formulations in elastodynamics

3.1 Introduction

The general equations governing linear scalar wave propagation, linear elastodynamics and their boundary integral formulations are briefly introduced in this chapter. Section 3.2 is devoted to the governing equations of the scalar wave problem and linear elastodynamics. The Green's identity and reciprocal relation, which lay the mathematical foundation of BEM, are described in Section 3.3 . Green's functions and their properties for scalar wave propagation and linear elastodynamics are given in Section 3.4. The reciprocal relationship derived in Section 3.3 leads to the integral representation of the general problem of wave propagation and elastodynamic problems (Section 3.5). Finally, the integral equations for the wave propagation and elastodynamic problems are derived (Section 3.6). Only basic equations and ideas needed to develop the boundary element solution are presented here. More complete details can be found in the books by Manolis & Beskos [51], Dominguez [20] and C Pozrikidis [68].

3.2 Governing equations

The governing equation of scalar wave propagation

For a regular domain Ω with boundary Γ, the governing equation for scalar wave propagation with non-zero body sources in the region Ω is:

$$\frac{\partial^2 \mathbf{u}(\mathbf{x},t)}{\partial x^2} + \frac{\partial^2 \mathbf{u}(\mathbf{x},t)}{\partial y^2} + \frac{\partial^2 \mathbf{u}(\mathbf{x},t)}{\partial z^2} + \frac{1}{c^2}\boldsymbol{b} = \frac{1}{c^2}\frac{\partial^2 \mathbf{u}(\mathbf{x},t)}{\partial t^2} \tag{3.1}$$

where $\mathbf{u}(\mathbf{x},t)$ is the displacement in 4D space-time, $\mathbf{x}$ denotes the spatial coordinates (x,y,z), $\boldsymbol{u}_{,ii}$ is a tensor notation for $\bigtriangledown^2\mathbf{u} = \frac{\partial u^2}{\partial x^2} + \frac{\partial u^2}{\partial y^2} + \frac{\partial u^2}{\partial z^2}$, $\mathbf{b}$ is the body force and c is the wave speed.

Alternatively, in vector form:

$$\boldsymbol{\nabla}^2\mathbf{u} + \frac{1}{c^2}\boldsymbol{b} = \frac{\partial^2 \mathbf{u}}{\partial t^2} \tag{3.2}$$

The boundary conditions are:

$$\mathbf{u}(\mathbf{x},t) = \bar{\mathbf{u}}(\mathbf{x},t) \qquad on \qquad \Gamma_1$$

$$\frac{\partial \mathbf{u}(\mathbf{x},t)}{\partial \mathbf{n}(\mathbf{x})} = \bar{q}(\mathbf{x},t) \qquad on \qquad \Gamma_2 \tag{3.3}$$

where $\mathbf{n}$ is the unit normal vector at $\mathbf{x}$, $\Gamma_1 \bigcup \Gamma_2 = \Gamma$ and $\Gamma_1 \bigcap \Gamma_2 = \emptyset$.

The initial conditions are:

$$\mathbf{u}(\mathbf{x},0) = \mathbf{u}_0(\mathbf{x})$$

and

$$\frac{\partial \mathbf{u}(\mathbf{x},0)}{\partial t} = \mathbf{v}_0(\mathbf{x}) \tag{3.4}$$

The governing equation of linear elastodynamics

The governing equations of elastodynamics are:

$$\mu \boldsymbol{u}_{i,jj} + (\lambda+\mu)\mathbf{u}_{j,ji} + \boldsymbol{\rho}\mathbf{b}_i = \frac{\partial^2 \mathbf{u}_i}{\partial t^2} \tag{3.5}$$

where $\mathbf{u}(\mathbf{x},t)$ is the displacement in 4D space-time, $\mathbf{p}(\mathbf{x},t)$ are tractions on the boundary, $\boldsymbol{x}$ denotes the spatial coordinates (x,y,z), $\mathbf{b}_j$ is the body force, ρ is the density, μ and λ are Lame's constants.

Alternatively, in vector form:

$$\mu \bigtriangledown \boldsymbol{u}^2 + (\lambda+\mu) \bigtriangledown \bigtriangledown \cdot \mathbf{u} + \boldsymbol{\rho}\mathbf{b} = \frac{\partial^2 \mathbf{u}}{\partial t^2} \tag{3.6}$$

The displacements $\mathbf{u}$ are prescribed on the Γ_1, and the tractions $\mathbf{p}$ are prescribed on the Γ_2, *thus:*

$$\mathbf{u}_i(\boldsymbol{x},t)\,|_{\Gamma_1} = \overline{\mathbf{u}}_i(\boldsymbol{x},t) \qquad \mathbf{p}_i(\boldsymbol{x},t) = |_{\Gamma_2}\,\overline{\mathbf{p}}_i(\boldsymbol{x},t) \tag{3.7}$$

To complete the problem statement, initial conditions are prescribed at time $t=0$:

$$\mathbf{u}_i(\boldsymbol{x},0)\,|_{\Gamma} = \mathbf{u}_{0i}(\boldsymbol{x}) \qquad \dot{\mathbf{u}}_i(\boldsymbol{x},0)\,|_{\Gamma} = \mathbf{v}_{0i}(\boldsymbol{x}) \tag{3.8}$$

for all the points $\mathbf{x}$ in the body Ω, where the boundary $\Gamma_1 \bigcup \Gamma_2 = \Gamma$ and $\Gamma_1 \bigcap \Gamma_2 = \emptyset$.

The Lame's constants can be written in terms of Young's modulus E and Poisson's ratio ν

$$\mu = \frac{E}{2(1+v)} \quad ; \quad \lambda = \frac{E}{(1+\nu)(1-2\nu)} \tag{3.9}$$

3.3 Green's identities and reciprocal relations

3.3.1 Green's identities and reciprocal relations for Laplace equation

Green's first identity states that any two twice-differentiable functions $f(x,y,z)$ and $\phi(x,y,z)$ satisfy the relation:

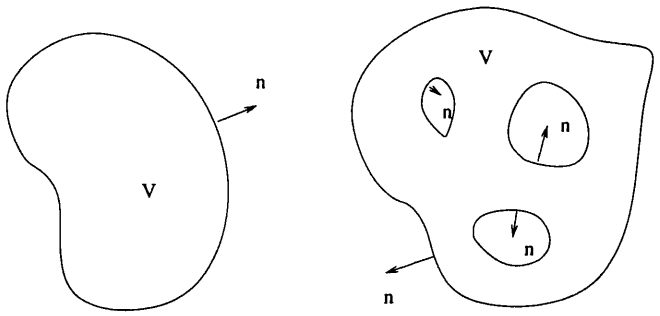


Figure 3.1: A control volume (a) with a closed surface (b) with a collection of closed surface

$$\phi \bigtriangledown^2 f = \bigtriangledown \cdot (\phi \bigtriangledown f) - \bigtriangledown\phi \cdot \bigtriangledown f \tag{3.10}$$

In index notation, this can be written as:

$$\frac{\partial^2 f}{\partial x_i \partial x_i} = \frac{\partial f}{\partial x_i}(\phi \frac{\partial f}{\partial x_i}) - \frac{\partial \phi}{\partial x_i}\frac{\partial f}{\partial x_i} \tag{3.11}$$

Interchanging the roles of f and ϕ, we obtain:

$$f \bigtriangledown^2 \phi = \bigtriangledown \cdot (f \bigtriangledown \phi) - \bigtriangledown f \cdot \bigtriangledown \phi \tag{3.12}$$

Subtracting 3.10 from 3.12, we derive Green's second identity:

$$\phi \bigtriangledown^2 f - f \bigtriangledown^2 \phi = \bigtriangledown \cdot (\phi \bigtriangledown f - f \bigtriangledown \phi) \tag{3.13}$$

If both functions f and ϕ satisfy Laplace's equation, the left-hand side of Eq. 3.13 vanishes, and we get the reciprocal relation for the Laplacian operator:

$$\bigtriangledown \cdot (\phi \bigtriangledown f - f \bigtriangledown \phi) = 0 \tag{3.14}$$

Integral form of the reciprocal relation

We integrate both sides of Eq. 3.13 over an arbitrary volume V which is bounded by a closed surface or a collection of closed surfaces Γ, as illustrated in Fig. 3.1.

Using the Gauss divergence theorem, the volume integral can be converted to a

surface integral, and we obtain the integral form of the reciprocal relation:

$$\int_V \nabla \cdot (\phi \nabla f - f \nabla \phi) dV = \int_\Gamma n \cdot (\phi \nabla f - f \nabla \phi) d\Gamma = 0 \tag{3.15}$$

or

$$\int_\Gamma n \cdot \phi \nabla f d\Gamma = \int_\Gamma n \cdot f \nabla \phi d\Gamma \tag{3.16}$$

3.3.2 Riemann convolution and dynamic reciprocal relation

The reciprocal relation in elastodynamics is an extension of the reciprocal relation for Laplacian operators. Before deriving it, some properties of the Riemann convolution are introduced at first. We assume two scalar functions $\phi(\mathbf{x}, t)$ and $f(\mathbf{x}, t)$ are continuous in the space $\Omega \times T^+$; where Ω is a region, T^- is the open time interval $(-\infty, 0)$ and T^+ is the half open time interval $[0, \infty)$. The Riemann convolution is defined as:

$$\phi \star f = \int_0^t \phi(\mathbf{x}, t-\tau) f(\mathbf{x}, \tau) d\tau \qquad \forall (\mathbf{x}, t) \in \Omega \times T^+$$

$$\phi \star f = 0 \qquad \forall (\mathbf{x}, t) \in \Omega \times T^- \tag{3.17}$$

Some useful properties of Riemann convolution are:

$$\phi \star f = f \star \phi$$

$$\frac{\partial}{\partial t}(\phi \star f) = \frac{\partial \phi}{\partial t} \star f + \phi(\mathbf{x}, 0) f(\mathbf{x}, t) \tag{3.18}$$

As a direct corollary from these properties, we can derive the second time derivative of the Riemann convolution:

$$\frac{\partial^2}{\partial t^2}(\phi \star f) = \frac{\partial^2 \phi}{\partial t^2} \star f + \frac{\partial \phi}{\partial t}(\mathbf{x}, 0) f + \phi(\mathbf{x}, 0) \frac{\partial f}{\partial t} \tag{3.19}$$

The reciprocal of the above formula is:

$$\frac{\partial^2}{\partial t^2}(f \star \phi) = \frac{\partial^2 f}{\partial t^2} \star \phi + \frac{\partial f}{\partial t}(\mathbf{x}, 0) \phi + f(\mathbf{x}, 0) \frac{\partial \phi}{\partial t} \tag{3.20}$$

Subtracting Eq. 3.19 from Eq. 3.20, and noting that the left sides are the identical, we obtain:

$$\frac{\partial^2 \phi}{\partial t^2} \star f = \frac{\partial^2 f}{\partial t^2} \star \phi + \frac{\partial f}{\partial t}(x, 0) \phi + f(\mathbf{x}, 0) \frac{\partial \phi}{\partial t} - \frac{\partial \phi}{\partial t}(\mathbf{x}, 0) f - \phi(\mathbf{x}, 0) \frac{\partial f}{\partial t} \tag{3.21}$$

3.3.3 Reciprocal relation for wave equation and elastodynamics

From the reciprocal relations for the Laplacian operator Eq. 3.14 and the reciprocal relations for the second time derivative (Eq. 3.21), we can derive the reciprocal relation for the wave equation. Assume that any two twice-differentiable functions $f(\mathbf{x}, t)$ and $\phi(\mathbf{x}, t)$ both satisfy the wave equation, then

$$\phi \star (\nabla^2 f + \frac{1}{c^2} \mathbf{b} - \frac{1}{c^2} \frac{\partial^2 f}{\partial t^2}) = 0 \tag{3.22}$$

Interchanging the roles of f and ϕ , we obtain

$$f \star (\nabla^2 \phi + \frac{1}{c^2} \mathbf{b}^* - \frac{1}{c^2} \frac{\partial^2 \phi}{\partial t^2}) = 0 \tag{3.23}$$

Subtracting 3.23 from 3.22, and using Eq. 3.14 and 3.21, we have:

$$\nabla\cdot(\phi\nabla f-f\nabla\phi)+\frac{1}{c^2}(b\star\phi-\frac{\partial\phi}{\partial t}(\mathbf{x},0)f-\phi(\mathbf{x},0)\frac{\partial f}{\partial t}+\frac{\partial f}{\partial t}(\mathbf{x},0)\phi+f(\mathbf{x},0)\frac{\partial\phi}{\partial t}-b^*\star f)=0 \tag{3.24}$$

Eq. 3.24 can be integrated over an arbitrary volume V bounded by a closed surface Γ, using Gauss divergence theorem, leading to a surface integral. The integral form of the reciprocal relation for the wave equation is:

$$\begin{aligned}&\int_S \phi n\cdot\nabla f dS+\int_V\frac{1}{c^2}(b\star\phi+\frac{\partial f}{\partial t}(\mathbf{x},0)\phi+f(\mathbf{x},0)\frac{\partial\phi}{\partial t})d\Omega \\ =\ &\int_S fn\cdot\nabla\phi dS+\int_V\frac{1}{c^2}(b^*\star f+\frac{\partial\phi}{\partial t}(\mathbf{x},0)f+\phi(\mathbf{x},0)\frac{\partial f}{\partial t})d\Omega\end{aligned} \tag{3.25}$$

The integral form of the reciprocal relation for elastodynamics can be derived in a similar way. The only difference is that ϕ_k and f_k are vector functions instead of scalar functions.

$$\begin{aligned}&\int_S \phi_k n\cdot\nabla f_k dS+\int_V\rho(\boldsymbol{b}_k\star\phi_k+\frac{\partial f_k}{\partial t}(\mathbf{x},0)\phi_k+f_k(\mathbf{x},0)\frac{\partial\phi_k}{\partial t})d\Omega \\ &\int_S f_k n\cdot\nabla\phi_k dS+\int_V\frac{1}{c^2}(b_k^*\star f_k+\frac{\partial\phi_k}{\partial t}(\mathbf{x},0)f_k+\phi_k(\mathbf{x},0)\frac{\partial f_k}{\partial t})d\Omega\end{aligned} \tag{3.26}$$

3.4 Green's Functions

Green's functions are fundamental solutions of the governing PDEs when the body force is replaced by a unit point load. The difference between FEM and BEM is that FEM uses piece-wise polynomials as trial functions while BEM uses Green's functions to do so. The Green's functions can also be considered to the counterpart of the inverse of the stiffness matrix in FEM with infinite dimensions.

3.4.1 Green's functions for 3D scalar waves

By definition, the Green's function for the wave equation in 3D satisfies the singularly forced wave equation [68]:

$$u_{,ii} + \delta(\mathbf{x} - \mathbf{x}_0)\delta(t) = \frac{1}{c^2}\frac{\partial^2 \mathbf{u}}{\partial t^2} \tag{3.27}$$

where

- $\mathbf{x} = (x, y, z)$ are the coordinates of the " field point ".
- $\mathbf{x}_0 = (x_0, y_0 z_0)$ are the coordinates of the " source point ", or "singular point ".
- $\delta(\mathbf{x} - \mathbf{x}_0)$, explicitly written as $\delta(x - x_0, y - y_0, z - z_0)$, is the Dirac delta function in 3D, and $\delta(t)$ is the Dirac delta function in the time domain.

The Dirac delta function in 3D has the following properties:

- $\delta(x - x_0, y - y_0, z - z_0)$ is zero everywhere except at the point $x = x_0, y = y_0, z = z_0$, where it becomes infinite.
- The integral of the Dirac delta function in 3D over a volume Ω that contain the singular point $\mathbf{x}_0 = (x_0, y_0, z_0)$ is equal to unity,

$$\int_\Omega \delta(x - x_0, y - y_0, z - z_0)dxdydz = 1 \tag{3.28}$$

- The integral of the product of an arbitrary function $f(x, y, z)$ and the Dirac delta function in 3D over a volume Ω that contains the singular point $\mathbf{x}_0 = (x_0, y_0 z_0)$ is equal to the value of the function at the singular point. If the volume Ω does not contain the singular point $\mathbf{x}_0 = (x_0, y_0 z_0)$, the value of integral is zero.

$$\int_\Omega \delta(x - x_0, y - y_0, z - z_0) f(x, y, z)dxdydz = f(x_0, y_0 z_0) \tag{3.29}$$

- The Dirac Delta function $\delta(x - x_0, y - y_0, z - z_0)$ arises from the function:

$$F(r) = (\frac{\lambda}{\pi L^2})^{\frac{3}{2}} exp(-\lambda \frac{r^2}{L^2})$$

in the limit as the dimensionless parameter λ tends to infinity, where

$$r = |\mathbf{x} - \mathbf{x}_0| = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2}$$

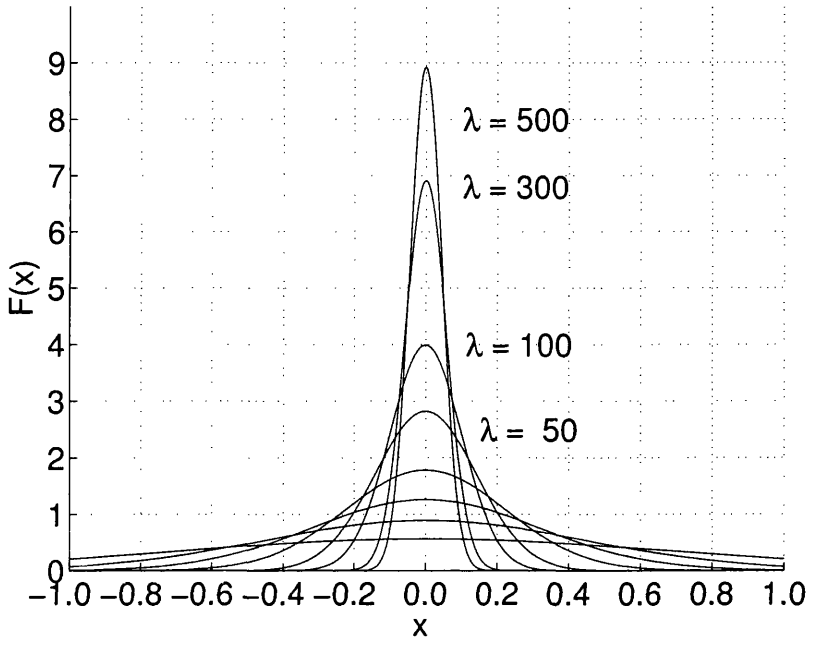


Figure 3.2: A series of functions which approximate the Dirac delta function

and L is an arbitrary length.

Fig. 3.2 shows a series of functions F plotted against the distance $\mathbf{X} = \mathbf{x} - \mathbf{x}_0$ for $\lambda = 2, 5, 10, 20, 50, 100, 300, 500$. The maximum height of each graph is inversely proportional to its width, so that the area underneath each graph is equal to unity.

3.4.1.1 The free space Green's function

The Green's function for an infinite open domain is [68]:

$$G(\mathbf{x}, t, \mathbf{x}_0) = \frac{1}{4\pi r}\delta(t - \frac{r}{c}) = \frac{c}{4\pi r}\delta(ct - r) \tag{3.30}$$

where

$$r = |\mathbf{x} - \mathbf{x}_0| = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2}$$

3.4.1.2 Green's function in bounded domains

The Green's function for bounded domains is composed of a singular part which is the same as the free-space Green's function (Eq. 3.30), and a complementary part which may be written as a harmonic function H, which is non-singular throughout the solution domain [68]:

$$G(\mathbf{x}, t, \mathbf{x}_0) = \frac{1}{4\pi r}\delta(t - \frac{r}{c}) + H(\mathbf{x}, \mathbf{x}_0) \tag{3.31}$$

In general, the particular form of H depends on the geometry of the boundary. For certain simple boundary geometries involving planar and spherical shapes, the function H can be found by the method of images, i.e. by placing free space Green's functions and their derivatives at specially chosen positions outside the solution domain. For example, the Green's function for a half-space domain bounded by a plane surface located at $x = a$ is:

$$G(x, t, x_0) = \frac{1}{4\pi r}\delta(t - \frac{r}{c}) + \frac{1}{4\pi r^{Im}}\delta(t - \frac{r^{Im}}{c}) \tag{3.32}$$

where $r = |\mathbf{x} - \mathbf{x}_0|$, $r^{Im} = |\mathbf{x} - \mathbf{x}_0^{Im}|$ and $\mathbf{x}_0^{Im} = (2a - x_0, {}_{00})$ is the image of the singular points $\mathbf{x}_0$ with respect to the surface.

3.4.1.3 Integral properties of Green's functions

We consider a single or multiple connected control volume Ω in 3D, bounded by a collection of closed surfaces Γ. We assume that all surfaces are smooth, i.e. they do not exhibit conical, edge-like or cusp-like corners.

Integrating Eq. 3.30 over the control volume Ω, and using Gauss divergence theorem, and the properties of Dirac delta function, we obtain the integral identity:

$$\int_\Omega \frac{\mathbf{1}}{\mathbf{c^2}}\frac{\partial^2 \mathbf{u}}{\partial t^2} dV - \int_\Gamma \mathbf{n} \cdot \bigtriangledown(\mathbf{x}, t, \mathbf{x}_0) d\Gamma = \begin{cases} 1 & \textit{when } \mathbf{x}_0 \textit{ is inside } \Omega \\ \frac{1}{2} & \textit{when } \mathbf{x}_0 \textit{ is on } \Gamma \\ 0 & \textit{when } \mathbf{x}_0 \textit{ is outside } \Omega \end{cases} \tag{3.33}$$

where the unit normal vector $\mathbf{n}$ points outwards from the control volume Ω. When the point x_0 is located on the boundary Γ, the integral on the left-hand side of Eq. 3.33 is integrated as a principal value integral.

3.4.1.4 Green's function dipole

Differentiating a Green's function with respect to the coordinates of the source point $\mathbf{x}_0 = (x_0, y_0 z_0)$, we obtain a vectorial singularity called the Green's function dipole.

$$G^D = \nabla_0 G(x,t,x_0) = \left[\frac{\partial G(x,t,x_0)}{\partial x_0}, \frac{\partial G(x,t,x_0)}{\partial y_0}, \frac{\partial G(x,t,x_0)}{\partial z_0}\right] \tag{3.34}$$

where the subscript " 0 " means differentiation with respect to the Cartesian components of x_0.

$$\Phi(x,t,x_0) = \mathbf{d} \cdot \nabla_0 G(x,t,x_0) \tag{3.35}$$

where $\Phi(x,t,x_0)$ is the dipole field function which describes the field direction and strength of a dipole. Vector $\mathbf{d}$ is the directional distance between two source points. The scalar field of a dipole may represent the temperature or electric concentration field due to a point source dipole of heat or electrical charge located at the point x_0. The direction and strength of the dipole is determined by the orientation and magnitude of the vector d.

Differentiating the right-hand side of Eq. 3.30, we get the 3D free space point source dipole

$$G_i^D = \frac{\hat{x}_i}{4\pi r^3}\delta(t - \frac{r}{c}) + \frac{\hat{x}_i}{4\pi r^2}\frac{\partial \delta(t - \frac{r}{c})}{\partial r} \tag{3.36}$$

where $i = 1, 2, 3$ corresponds to x, y, z, $\hat{\mathbf{x}} = \mathbf{x} - \mathbf{x}_0$ and $r = |\mathbf{x} - \mathbf{x}_0|$.

3.4.2 Green's functions for 3D elastodynamics

The Green's function for 3D elastodynamics is the solution of the following equation:

$$\rho c_1^2 \boldsymbol{u}_{i,jj} + \rho c_2^2 + \delta(\mathbf{x} - \mathbf{x}_0)\delta(t)\, e_i = \frac{\partial^2 \mathbf{u}_i}{\partial t^2} \tag{3.37}$$

where $\boldsymbol{u}$ is the displacement, c_1 is the dilatational wave speed and c_2 is the shear wave speed, and e_i is an unit direction vector, which points in the x, y, z direction when $i = 1,\ 2,\ 3$.

3.4.2.1 The free-space Green's function

The free-space Green's function $G_{lk}(\boldsymbol{x}, t; \boldsymbol{x}^i)$ for an infinite domain is: (Dominguez [20])

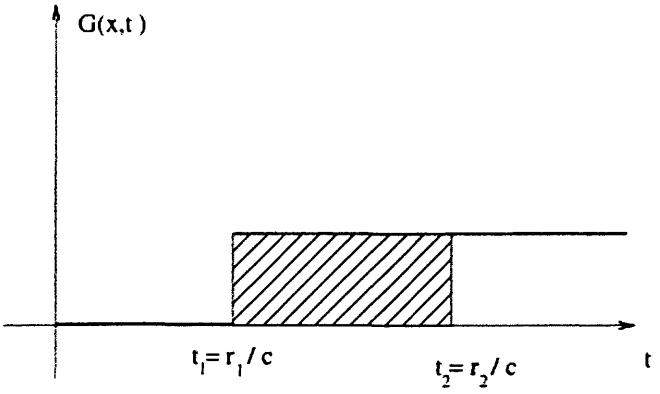


Figure 3.3: Impulses received at $t = |\mathbf{x} - \mathbf{x}_0|/c_1$ and $t = |\mathbf{x} - \mathbf{x}_0|/c_2$

$$\begin{aligned} G_{lk}\left(\boldsymbol{x}, t; \boldsymbol{x}^i\right) &= \frac{1}{4\pi\rho}\{\frac{t}{r^2}\left(\frac{3r_{,l}r_{,k}}{r} - \frac{\delta_{lk}}{r}\right)[H\left(t - \frac{r}{c_1}\right) - H\left(t - \frac{r}{c_2}\right)]\} \\ &+ \frac{1}{4\pi\rho}\{\frac{r_{,l}r_{,k}}{r}[\frac{1}{c_1^2}\delta(t - \frac{r}{c_1}) - \frac{1}{c_2^2}\delta(t - \frac{r}{c_2})] \\ &+ \frac{\delta_{lk}}{rc_2^2}\delta(t - \frac{r}{c_2})\} \end{aligned} \tag{3.38}$$

where $r = |\boldsymbol{x} - \boldsymbol{x}^i|$, $H(\mathbf{x})$ is the Heaviside function and $\delta(\mathbf{x})$ is the Dirac-delta function. Any impulse emitted from $\mathbf{x}_0$ at $t = 0$ is felt at $\mathbf{x}$ only from the arrival of the first wave ($t = |\mathbf{x} - \mathbf{x}_0|/c_1$) up to the arrival of the second wave ($t = |\mathbf{x} - \mathbf{x}_0|/c_2$), as shown in the Fig. 3.3.

3.4.2.2 Free-space stress tensor and tractions tensor

The corresponding stress tensor is derived from Eq. 3.38 using the constitutive relation:

$$\sigma_{km} = \rho(\mathrm{c}_1^2 - \mathrm{c}_2^2)\mathrm{u}_{\mathrm{j,j}}\delta_{\mathrm{km}} + \rho\mathrm{c}_2^2(\mathrm{u}_{\mathrm{k,m}} + \mathrm{u}_{\mathrm{m,k}}) \tag{3.39}$$

We also have

$$\sigma_{km} = \sigma_{lkm}e_l \tag{3.40}$$

where σ_{lkm} represent components of the stress tensor in the k and m direction when a point load is applied in the l direction. From Eq. 3.38, Eq. 3.39 and Eq.

3.40, we obtain:

$$\begin{aligned}\sigma^*_{lkm} = & \frac{1}{4\pi}\{-6c_2^2[5\frac{r_{,l}r_{,k}r_{,m}}{r^2} - \frac{\delta_{km}r_{,l}+\delta_{kl}r_{,m}+\delta_{kl}r_{,k}}{r^2}]\cdot\frac{t}{r^2}[H(t-\frac{r}{c_1})-H(t-\frac{r}{c_2})] \\ + & 2[\frac{6}{r^2}r_{,l}r_{,k}r_{,m} - \frac{\delta_{km}r_{,l}+\delta_{kl}r_{,m}+\delta_{lm}r_{,k}}{r^2}]\cdot[\delta(t-\frac{r}{c_2})-\frac{c_2^2}{c_1^2}\delta(t-\frac{r}{c_1})] \\ + & \frac{2}{rc_2}r_{,l}r_{,k}r_{,m}[\dot\delta(t-\frac{r}{c_2})-\frac{c_2^3}{c_1^3}\dot\delta(t-\frac{r}{c_1})] - \frac{\delta_{km}r_{,l}}{r^2}(1-2\frac{c_2^2}{c_1^2})[\delta(t-\frac{r}{c_1})-\frac{r}{c_1}\dot\delta(t-\frac{r}{c_1})] \\ - & \frac{\delta_{kl}r_{,m}+\delta_{ml}r_{,k}}{r^2}[\delta(t-\frac{r}{c_2})-\frac{r}{c_2}\dot\delta(t-\frac{r}{c_2})] \end{aligned} \tag{3.41}$$

where where σ^*_{lkm} represent components of the stress tensor in the the k and m direction when a Dirac Delta function load is applied in the l direction. The corresponding traction kernel function $p^*_{lk}(\boldsymbol{x},t;\boldsymbol{x_0})$, (when l indicates the traction in the direction l , and k indicate the point force direction k), can be obtained from Eq. 3.41, using equilibrium equation, which yields:

$$p^*_{lk}(\boldsymbol{x},t;\boldsymbol{x}^i) = \frac{1}{4\pi}\{\left(\frac{\partial r}{\partial n}\delta_{lk}+r_{,k}n_l\right)A + r_{,l}r_{,k}\frac{\partial r}{\partial n}B + r_{,l}n_kC\} \tag{3.42}$$

where the coefficients A, B, C are:

$$\begin{aligned}A = & \frac{6c_2^2}{r^4}t\,[H(t-\frac{r}{c_1})-H(t-\frac{r}{c_2})] - \frac{2}{r^2}[[\delta(t-\frac{r}{c_2})-\frac{c_2^2}{c_1^2}\delta(t-\frac{r}{c_1})] \\ - & \frac{1}{r^2}[\delta(t-\frac{r}{c_2})-\frac{r}{c_2}\dot\delta(t-\frac{r}{c_2})]\end{aligned}$$

$$\begin{aligned}B = & \frac{-30c_2^2}{r^4}t\,[H(t-\frac{r}{c_1})-H(t-\frac{r}{c_2})] + \frac{12}{r^2}[[\delta(t-\frac{r}{c_2})-\frac{c_2^2}{c_1^2}\delta(t-\frac{r}{c_1})] \\ - & \frac{2}{rc_2}[\dot\delta(t-\frac{r}{c_2})-\frac{c_2^3}{c_1^3}\dot\delta(t-\frac{r}{c_1})]\end{aligned}$$

$$\begin{aligned}C = & \frac{6c_2^2}{r^4}t\,[H(t-\frac{r}{c_1})-H(t-\frac{r}{c_2})] - \frac{2}{r^2}[[\delta(t-\frac{r}{c_2})-\frac{c_2^2}{c_1^2}\delta(t-\frac{r}{c_1})] \\ - & \frac{1}{r^2}[1-2\frac{c_2^2}{c_1^2}][\delta(t-\frac{r}{c_{21}})-\frac{r}{c_1}\dot\delta(t-\frac{r}{c_1})]\end{aligned} \tag{3.43}$$

3.5 Integral representation

3.5.1 For Green's functions

To develop the boundary integral representation, we apply Green's second identity in 4D space-time (Eq. 3.24), and recast it in the form:

$$\frac{1}{c^2}(-b\star\phi+\frac{\partial\phi}{\partial t}(\mathbf{x},0)f+\phi(\mathbf{x},0)\frac{\partial f}{\partial t}-\frac{\partial f}{\partial t}(\mathbf{x},0)\phi-f(\mathbf{x},0)\frac{\partial\phi}{\partial t}+b^*\star f)=\nabla\cdot(\phi\nabla f-f\nabla\phi) \tag{3.44}$$

Next, we assume that f is a solution of the wave or elastodynamic equation, and ϕ is assumed to the Green's function G of the same governing equation, namely the solution of the wave equation or elastodynamic equation in the infinite domain for a unit impulse point force as:

$$\frac{1}{c^2}b^*=\delta(\mathbf{x}-\mathbf{x_0})\delta(t) \tag{3.45}$$

centered at the point $\mathbf{x}_0$ at $t=0$. Using Eq. 3.44, we obtain:

$$\begin{aligned}\frac{1}{c^2}(-b\star\phi+\frac{\partial\phi}{\partial t}(\mathbf{x},0)f+\phi(\mathbf{x},0)\frac{\partial f}{\partial t}-\frac{\partial f}{\partial t}(\mathbf{x},0)\phi-f(\mathbf{x},0)\frac{\partial\phi}{\partial t})&\\ +\delta(\mathbf{x}-\mathbf{x}_0)\delta(t)\star f=\nabla\cdot(\phi\nabla f-f\nabla\phi)&\end{aligned} \tag{3.46}$$

We now select a control volume V bounded by a closed surface Γ, integrate Eq. 3.44 (using Eq. 3.45) over the control volume and time from the initial instant $t=0$ up to the current instant t_0^+ (the plus sign here means the upward limit). Applying Gauss divergence theorem to convert the volume integral to a surface integral on

right-hand side, we obtain:

$$\begin{aligned}
&\int_0^{t_0^+}\int_V \delta(\mathbf{x}-\mathbf{x}_0)\delta(t)\star f dV dt + \int_0^{t_0^+}\int_V \frac{1}{c^2}(\frac{\partial G}{\partial t}(\mathbf{x},0)f(\mathbf{x},t) + G(\mathbf{x},0)\frac{\partial f(\mathbf{x},t)}{\partial t})dV dt \\
= &\int_0^{t_0^+}\int_\Gamma G(\mathbf{x},t,\mathbf{x}_0)[n\cdot\bigtriangledown f(\mathbf{x},t)] - f(\mathbf{x},t)[n\cdot\bigtriangledown G(\mathbf{x},t,\mathbf{x}_0)]d\Gamma dt \\
+ &\int_0^{t_0^+}\int_V \frac{1}{c^2}(-b\star G(\mathbf{x},t,\mathbf{x}_0))dV dt \\
+ &\int_0^{t_0^+}\int_V \frac{1}{c^2}(\frac{\partial f}{\partial t}(\mathbf{x},0)G(\mathbf{x},t,\mathbf{x}_0) + f(\mathbf{x},0)\frac{\partial G(\mathbf{x},t,\mathbf{x}_0)}{\partial t}))dV dt
\end{aligned} \tag{3.47}$$

Using the "*sifting*" property of the delta function $\delta(t-t_0)$ and $\delta(\mathbf{x}-\mathbf{x}_0)$, and noting that $G(\mathbf{x},0)$ and $\frac{\partial G}{\partial t}(\mathbf{x},0)$ are both zero since Green's function G remains quiescent at $t=0$, we obtain the integral representation:

$$\begin{aligned}
f(x_0,t) = &\int_0^{t_0^+}\int_\Gamma G(x,t,x_0)[n\cdot\bigtriangledown f(x,t)]d\Gamma dt - \int_0^{t_0^+}\int_\Gamma f(x,t)[n\cdot\bigtriangledown G(x,t,x_0)]d\Gamma dt \\
+ &\int_0^{t_0^+}\int_V\int_V \frac{1}{c^2}(-b\star G(\mathbf{x},t,\mathbf{x}_0))dV dt \\
+ &\int_0^{t_0^+}\int_V \frac{1}{c^2}(\frac{\partial f}{\partial t}(\mathbf{x},0)G(\mathbf{x},t,\mathbf{x}_0) + f(\mathbf{x},0)\frac{\partial G(\mathbf{x},t,\mathbf{x}_0)}{\partial t}))dV dt
\end{aligned} \tag{3.48}$$

In this equation, there are:

- Two domain integrals: one involves the product of the Green's function and the body force; another involves the initial conditions $f(x,0)$ and $\frac{\partial f}{\partial t}(x,0)$.
- Two boundary integrals: these integrals represent the field due to dynamic impulse point sources and point-source dipoles distributed over the boundaries of the solution in the space-time domain. The analogy can be made between these two quantities with corresponding boundary distributions of electric charges and charge dipoles in electromagnetics, called *single-layer* and *double-layer potentials.* The densities, defined as the strength per unit of surface area, of these potentials are equal to the boundary distribution of the normal derivative and to the boundary values of the displacement potential of the solution.

Integral representation of the gradient

An integral representation of the gradient ∇f can be derived by differentiating Eq. 3.48, which yields

$$\begin{aligned}\frac{\partial f(\mathbf{x}_0,t)}{\partial \mathbf{x}_{0i}} &= \int_0^{t_0^+}\int_\Gamma \frac{\partial G(\mathbf{x},t,\mathbf{x}_0)}{\partial \mathbf{x}_{0i}}[\mathbf{n}\cdot\nabla f(\mathbf{x},t)]d\Gamma dt - \int_0^{t_0^+}\int_\Gamma f(x,t)[\mathbf{n}_j\cdot\frac{\partial G(\mathbf{x},t,\mathbf{x}_0)}{\partial \mathbf{x}_{0i}\partial \mathbf{x}_j}]d\Gamma dt \\ &+ \int_0^{t_0^+}\int_V \frac{1}{c^2}(-b(\mathbf{x},t)\star\frac{\partial G(\mathbf{x},t,\mathbf{x}_0)}{\partial \mathbf{x}_{0i}})dVdt \\ &+ \int_0^{t_0^+}\int_V \frac{1}{c^2}(\frac{\partial f}{\partial t}(\mathbf{x},0)\frac{\partial G(\mathbf{x},t,\mathbf{x}_0)}{\partial \mathbf{x}_{0i}} + f(\mathbf{x},0)\frac{\partial G(\mathbf{x},t,\mathbf{x}_0)}{\partial \mathbf{x}_{0i}\partial t}))dVdt \end{aligned} \tag{3.49}$$

where in the term $[\mathbf{n}_j\cdot\frac{\partial G(\mathbf{x},t,\mathbf{x}_0)}{\partial \mathbf{x}_{0i}\partial \mathbf{x}_j}]$, summation over the repeated index j is implied.

3.5.2 For the wave equation

Substituting the free space Green's function for the 3D wave equation into the integral representation Eq. 3.48 for displacement u, the first time integral can be carried out, using the *"sifting"* property of Dirac delta function in time domain. Thus:

$$\begin{aligned}&\int_0^{t^+}\int_S G(x,t,x_0)[n\cdot\nabla f(x,t)]d\tau d\Gamma \\ =\ &\int_S \frac{1}{4\pi r}\int_0^{t^+}\delta[(t-\tau)-(r/c)]\cdot[n\cdot\nabla f(x,t)]d\tau d\Gamma(\mathbf{x}) \\ =\ &\int_S \frac{1}{4\pi r}\int_0^{t^+}\delta[t-(r/c)-\tau]\cdot[n\cdot\nabla f(x,t)]d\tau d\Gamma(\mathbf{x}) \\ =\ &\int_S \frac{1}{4\pi r}n\cdot\nabla f(x,t-(r/c))d\Gamma(\mathbf{x}) \end{aligned} \tag{3.50}$$

One can compute the normal derivative of the Green's function before computing the second term as follows:

$$\begin{aligned} n\cdot\nabla G(x,t,x_0) &= \frac{\partial G(x,t,x_0)}{\partial n} \\ &= \frac{\partial r}{\partial n}\{\frac{-1}{4\pi r^2}\delta[t-(r/c)-\tau]+\frac{1}{4\pi r}\frac{\partial}{\partial r}\delta[t-(r/c)-\tau]\} \\ &= \frac{\partial r}{\partial n}\{\frac{-1}{4\pi r^2}\delta[t-(r/c)-\tau]+\frac{1}{4\pi rc}\frac{\partial}{\partial \tau}\delta[t-(r/c)-\tau]\} \end{aligned} \tag{3.51}$$

Substituting Eq. 3.51 into the second integral in Eq. 3.50 yields:

$$\begin{aligned}
&\int_0^{t^+}\int_S f(x,t)[n\cdot\nabla G(x,t,x_0)]d\tau d\Gamma \\
&= \int_0^{t^+}\int_S \frac{\partial r}{\partial n}\{\frac{-1}{4\pi r^2}\delta[t-(r/c)-\tau]+\frac{1}{4\pi rc}\frac{\partial}{\partial\tau}\delta[t-(r/c)-\tau]\}\,u d\tau d\Gamma(\mathbf{x}) \\
&= -\frac{1}{4\pi}\int_S \frac{\partial r}{\partial n}\{\frac{1}{r^2}u(x,t-r/c)+\frac{1}{rc}[\frac{\partial u(x,\tau)}{\partial\tau}]_{\tau=t-(r/c)}\}\,d\Gamma(\mathbf{x}) \qquad (3.52)
\end{aligned}$$

where the property of the Dirac delta function:

$$\int_{-\infty}^{\infty}\frac{\partial^k\delta(x-a)}{\partial x^k}f(x)dx=(-1)^k[\frac{\partial^k f(x)}{\partial x^k}]_{x=a}$$

has been employed. Similarly the body force integral can be treated in the same way:

$$\begin{aligned}
\int_0^{t_0^+}\int_V \frac{1}{c^2}(b\star G(x,t,x_0))d\tau dV &= \int_V \frac{1}{4\pi r}\int_0^{t_0^+}\delta[t-(r/c)-\tau]b(x,t)d\tau dV \\
&= \frac{1}{4\pi}\int_V \frac{1}{r}b(x,t-r/c)dV \qquad (3.53)
\end{aligned}$$

Taking account of Eqs. 3.50, 3.52, 3.53, we get the integral representation for the wave equation:

$$\begin{aligned}
u(\boldsymbol{x}_0,t) &= \frac{1}{4\pi}\int_\Gamma \frac{1}{r}n\cdot\nabla f(\boldsymbol{x},t-(r/c))d\Gamma(\boldsymbol{x}) \\
&+ \frac{1}{4\pi}\int_\Gamma \frac{\partial r}{\partial n}\{\frac{1}{r^2}u(\boldsymbol{x},t-r/c)+\frac{1}{rc}[\frac{\partial u(\boldsymbol{x},\tau)}{\partial\tau}]_{\tau=t-(r/c)}\}\,d\Gamma(\boldsymbol{x}) \\
&+ \frac{1}{4\pi}\int_V \frac{1}{r}b(x,t-r/c)dtdV \qquad (3.54) \\
&+ \int_0^{t_0^+}\int_V \frac{1}{c^2}(\frac{\partial f}{\partial t}(\mathbf{x},0)\frac{\partial G(\mathbf{x},t,\mathbf{x}_0)}{\partial \mathbf{x}_{0i}}+f(\mathbf{x},0)\frac{\partial G(\mathbf{x},t,\mathbf{x}_0)}{\partial \mathbf{x}_{0i}\partial t}))dtdV
\end{aligned}$$

The above equation is also known as *Kirchhoff*'s integral representation.

3.5.3 For elastodynamics

From the reciprocal relation for the elastodynamics equation (Eq. 3.55):

$$\begin{aligned} & \int_\Gamma \phi_k n \cdot \bigtriangledown f_k d\Gamma + \int_V \rho(\boldsymbol{b}_k \star \phi_k + \frac{\partial f_k}{\partial t}(x,0)\phi_k + f_k(x,0)\frac{\partial \phi_k}{\partial t})d\Omega \\ = & \int_\Gamma f_k n \cdot \bigtriangledown \phi_k d\Gamma \\ + & \int_V \frac{1}{c^2}(b_k^* \star f_k + \frac{\partial \phi_k}{\partial t}(x,0)f_k + \phi_k(x,0)\frac{\partial f_k}{\partial t})d\Omega \end{aligned} \tag{3.55}$$

Assuming that ϕ_k is the free space Green's function $G(x,t,x_0)$ due to an impulse load applied at the internal point x_0 in the "l" direction at time $t=0$,

$$\rho b_k^* = \delta(x-x_0)\delta(t)\delta_{lk} \tag{3.56}$$

For quiescent initial conditions, Green's function $\phi_k(x,0)=0$ and $\frac{\partial \phi_k}{\partial t}(x,0)=0$, the integral representation for the displacement u_l at point (x_0,t):

$$\begin{aligned} u_l(x_0,t) = & \int_0^{t_0^+}\int_\Gamma G_{lk}(x,t-\tau,x_0)[n\cdot\bigtriangledown f_k(x,\tau)]d\Gamma d\tau \\ - & \int_0^{t_0^+}\int_\Gamma [n\cdot\bigtriangledown G_{lk}(x,t-\tau,x_0)]f_k(x,\tau)d\Gamma d\tau \\ + & \int_0^{t_0^+}\int_V \rho(-b_k(x,\tau)G_{lk}(x,t-\tau,x_0))dVd\tau \\ + & \int_0^{t_0^+}\int_V \rho[\frac{\partial f_k}{\partial t}(x,0)G_{lk}(x,t-\tau,x_0) + f_k(x,0)\frac{\partial G_{lk}(x,t-\tau,x_0)}{\partial t}]dVd\tau \end{aligned} \tag{3.57}$$

This equation gives the displacement u_l at point (x_0,t) in the terms of the displacements and tractions on the boundary Γ at time $\tau \in (0,t^+)$. The time integration for any point x starts when the first wave arrives at $\tau = t-|x-x_0|/c_1$and ends at $\tau = t-|x-x_0|/c_2$.

3.6 Boundary integral equations

To derive the integral equations for the boundary distributions of the displacement function u and its normal derivative, based on the integral representation Eq. 3.48, we take the limit as the field point x_0 approaches a local smooth boundary Γ.

$$\begin{aligned} f(x_0,t_0) &= \int_0^{t_0^+}\int_\Gamma G(x,t,x_0)[n\cdot\bigtriangledown f(x,t)]d\Gamma d\tau - \\ &- \int_0^{t_0^+}\int_\Gamma f(x,t)[n\cdot\bigtriangledown G(x,t,x_0)]d\Gamma d\tau \\ &+ \int_0^{t_0^+}\int_V \frac{1}{c^2}(-b(x,t)\star G(x,t,x_0))dVd\tau \\ &+ \int_0^{t_0^+}\int_V \frac{1}{c^2}(\frac{\partial f}{\partial t}(x,0)G(x,t,x_0)+f(x,0)\frac{\partial G(x,t,x_0)}{\partial t}))dVd\tau \end{aligned} \tag{3.58}$$

The first term on the right-hand side which involves the single-layer potential varies continuously as τ approaches the time t_0, and as $x_0$approaches and crosses the locally smooth boundary Γ. However, the second term which involves the double-layer potential produces a "*jump*" term. This can be evaluated by augmenting the boundary by a hemispherical surface around the collocation point x_0 and integrating Green's function over this spherical surface as its radius $\epsilon\to 0$, as shown in Fig. 3.4. For the dynamic problem, we need a *shrinking hemisphere* in space-time to compute the limit of the integral. This is the reason why we take the time convolution from 0 to t_0^+. Here t_0^+ means to take the upper limit of time t_0 to avoid boundary integral falling exactly on the time t_0, which allows us to compute the limit of the integral in time. The integrals are integrated as principal values:

$$\begin{aligned} &\int_0^{t_0^+}\int_{S-S_\epsilon} f(x,t)[n\cdot\bigtriangledown G(x,t,x_0)]dSd\tau \\ =\ &\lim_{\{x_0\to S,\tau\to t_0\}}\int_0^{t_0^+}\int_{S-S_\epsilon} f(x,t)[n\cdot\bigtriangledown G(x,t,x_0)]dSd\tau \\ +\ &\lim_{\{x_0\to S,\tau\to t_0\}}\int_0^{t_0^+}\int_{S_\epsilon} f(x,t)[n\cdot\bigtriangledown G(x,t,x_0)]dSd\tau \\ =\ &\int_0^{t_0^+}\int^{PV}_{S-S_\epsilon} f(x,t)[n\cdot\bigtriangledown G(x,t,x_0)]dSd\tau+\frac{1}{2}f(x_0,t_0) \end{aligned} \tag{3.59}$$

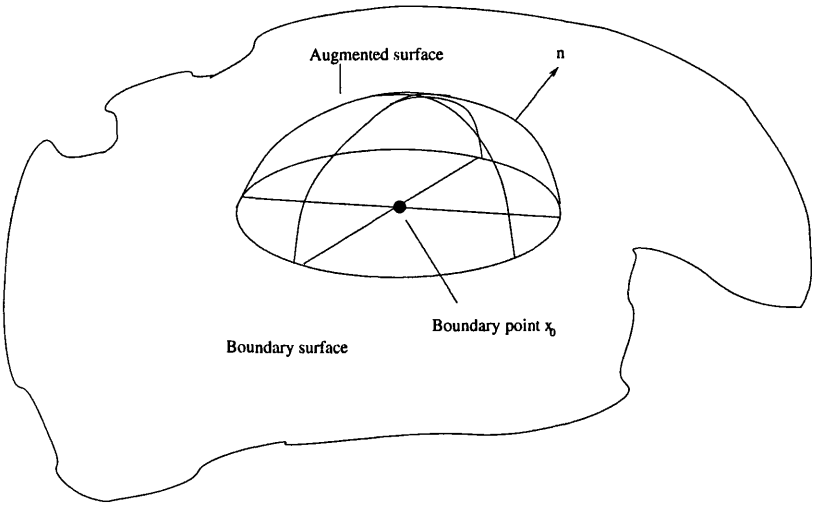


Figure 3.4: Boundary point augmented by a small hemisphere

where PV denotes the principal value integral.

Substituting Eq. 3.59 into Eq. 3.48, we obtain the integral equations for smooth boundaries.

$$\begin{aligned} c(x_0)f(x_0,t_0) &= \int_0^{t_0^+}\int_S G(x,t,x_0)[n\cdot\nabla f(x,t)]dSd\tau \\ &- \int_0^{t_0^+}\int_S^{PV} f(x,t)[n\cdot\nabla G(x,t,x_0)]dSd\tau \\ &+ \int_0^{t_0^+}\int_V \frac{1}{c^2}(-b(x,t)\star G(x,t,x_0))dVd\tau \\ &+ \int_0^{t_0^+}\int_V \frac{1}{c^2}(\frac{\partial f}{\partial t}(x,0)G(x,t,x_0)+f(x,0)\frac{\partial G(x,t,x_0)}{\partial t}))dVd\tau \end{aligned} \tag{3.60}$$

where the coefficient $c(x_0)=\frac{1}{2}$.

Boundary corners

If the boundary S is not smooth and it has edges, corners and other discontinuities in geometry, $c(x_0)$ is no longer equal to one half. The general formula is

$$c(x_0)=\frac{\alpha}{4\pi} \tag{3.61}$$

where the solid angle α is defined as the surface area of the sector of the unit sphere that is centered at the point x_0 and enclosed by the cone, as shown in the

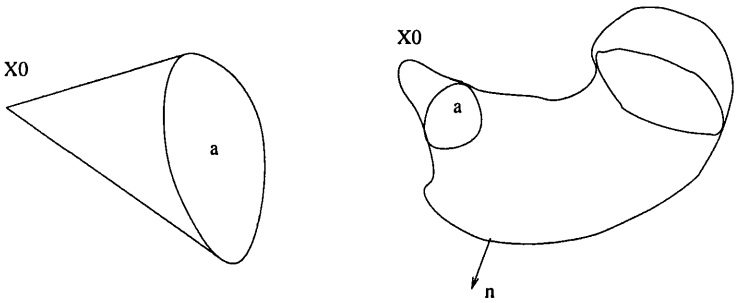


Figure 3.5: (a) a conical domain with apex at x_0 (b) solid angle α defined in a enclosed corner in a volume

Fig. 3.5.

Hypersingular integral equations

Integral equations for the boundary distribution of the normal derivative (flux or traction) can be derived from Eq. 3.60. Taking the limit as τ approaches time t_0, and x_0 approaches and crosses the locally smooth boundary Γ, differentiating Eq. 3.60, we obtain the hypersingular integral equations:

$$\begin{aligned} c(x_0)n_i(x_0)\frac{\partial f(x_0,t_0)}{\partial x_{0i}} &= \int_0^{t_0^+}\int_S^{PV} n_i(x_0)\frac{\partial G(x,t,x_0)}{\partial x_{0i}}[n\cdot\nabla f(x,t)]dSd\tau \\ &- \int_0^{t_0^+}\int_S^{FP} f(x,t)[n_i(x_0)\frac{\partial^2 G(x,t,x_0)}{\partial x_{0i}\partial x_j}\cdot n_j(x)]dSd\tau \\ &+ \int_0^{t_0^+}\int_V \frac{1}{c^2}(-b(x,t)\star n_i(x_0)\frac{\partial f(x_0,t_0)}{\partial x_{0i}})dVd\tau \\ &+ \int_0^{t_0^+}\int_V \frac{1}{c^2}(\frac{\partial f}{\partial t}(x,0)n_i(x_0)\frac{\partial f(x_0,t_0)}{\partial x_{0i}})dVd\tau \\ &+ \int_0^{t_0^+}\int_V \frac{1}{c^2}(f(x,0)n_i(x_0)\frac{\partial^2 G(x,t,x_0)}{\partial x_{0i}\partial t})dVd\tau \end{aligned} \tag{3.62}$$

where

$$\begin{aligned} &\int_0^{t_0^+}\int_S^{FP} f(x,t)[n_i(x_0)\frac{\partial^2 G(x,t,x_0)}{\partial x_{0i}\partial x_j}\cdot n_j(x)]dSd\tau \qquad (3.63)\\ &= \lim_{\{\varepsilon\to 0\}}\left\{\int_0^{t_0^+}\int_S f(x,t)[n_i(x_0)\frac{\partial^2 G(x,t,x_0)}{\partial x_{0i}\partial x_j}\cdot n_j(x)]dSd\tau - \frac{1}{2\varepsilon}f(x_0,t_0)\right\} \end{aligned}$$

FP denotes the finite part of the integral; ϵ is the radius of a small sphere centered at the point x_0, t_0 and excluded from the integration domain.

3.6.1 3D scalar wave

From the free-space Green's function for the 3D scalar wave equation (Eq. 3.30), the integral equation is:

$$\begin{aligned} c(x_0)u(x_0,t) &= \tfrac{1}{4\pi} \int_S \frac{1}{r} n \cdot \nabla u(x, t-(r/c)) dS(\mathbf{x}) \\ &+ \frac{1}{4\pi}\int_S \frac{\partial r}{\partial n}\{\frac{1}{r^2}u(x,t-r/c) + \frac{1}{rc}[\frac{\partial u(x,\tau)}{\partial \tau}]_{\tau=t-(r/c)}\}\, dS(\mathbf{x}) \\ &+ \frac{1}{4\pi}\int_V \frac{1}{r} b(x, t-r/c) dV \\ &+ \int_0^{t_0^+}\int_V \frac{1}{c^2}(\frac{\partial u}{\partial t}(x,0)G(x,t,x_0) + u(x,0)\frac{\partial G(x,t,x_0)}{\partial t}))dV \end{aligned} \tag{3.64}$$

where the integrals are integrated in the Cauchy Principal Value sense and the coefficient $c(x_0) = \frac{1}{2}$ when the boundary is smooth at x_0 and $c(x_0) = \frac{\alpha}{4\pi}$ when the boundary is not smooth, and α is the solid angle as defined in Eq. 3.61. Its values can also be calculated indirectly from the static boundary integral using constant potential considerations. It is also interesting to notice that the 3D scalar wave integral equation does not include the time integral. However, the same does not apply for 3D elastodynamics as shown below.

3.6.2 3D elastodynamics

From the free-space Green's function for 3D elastodynamics Eq. 3.38, we obtain the integral equation for the displacement u_l. However, since its mathematical expression is quite complicated (Dominguez [20]), here we omit its explicit expression:

$$\begin{aligned} c_{lk}(x_0)u_l(x_0,t) &= \int_0^{t_0^+}\int_S G_{lk}(x, t-\tau, x_0)[n \cdot \nabla f_k(x,\tau)]dS d\tau \\ &- \int_0^{t_0^+}\int_S [n \cdot \nabla G_{lk}(x, t-\tau, x_0)] f_k(x,\tau) dS d\tau \\ &+ \int_0^{t_0^+}\int_V \rho(G_{lk}(x, t-\tau, x_0) b_k(x,\tau)) dV d\tau \\ &+ \int_0^{t_0^+}\int_V \rho[(x, t-\tau, x_0)\frac{\partial f_k}{\partial t}(x,0)G_{lk} + \frac{\partial G_{lk}(x, t-\tau, x_0)}{\partial t} f_k(x,0)] dV d\tau \end{aligned} \tag{3.65}$$

where the coefficient $c_{lk}(x_0) = \frac{1}{2}\delta_{lk}$ when the boundary is smooth at x_0 and $c_{lk}(x_0) = \frac{\alpha}{4\pi\delta_{lk}}$ when the boundary is not smooth. α is the solid angle.

3.7 Summary

In this chapter, based on Green's identities and reciprocal relations for the 3D wave equation and elastodynamics , the integral representation formulas are derived. By taking the limit of these integrals when the source points approach the boundary, the boundary integral equations are obtained. Implementation of these equations will require the boundary discretization and shape functions to approximate the function distributions on the boundary: this will be described in the following chapters.

Chapter 4

Adaptive BEM for scalar wave propagation in 2D

4.1 Introduction

This chapter describes a new boundary element formulation for 2D scalar wave propagation based on an adaptive scheme in space-time (Section 4.2). Later a numerical implementation of the formulation derived above is presented (Section 4.3). This includes: spatial and temporal discretization, singular and non-singular kernel integrations, assembly of the system equations and solution algorithms. The error estimates for 2D wave BEM and adaptive schemes are presented in Section 4.4. Some examples of the application of the method are given to demonstrate its efficiency (Section 4.5).

4.2 Formulations of 2D scalar wave propagation in space-time

4.2.1 Governing equation

The scalar wave propagation equation in 2D is:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{1}{c^2}\mathrm{b} = \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} \qquad t > 0 \tag{4.1}$$

with initial conditions

$$u(\mathbf{x},0)\,|_{\Gamma} = u_0(\mathbf{x}) \qquad \frac{\partial u(\mathbf{x},0)}{\partial t}\,|_{\Gamma} = v_0(\mathbf{x}) \tag{4.2}$$

and boundary conditions

$$u(\mathbf{x},t)\,|_{\Gamma_1} = \overline{u}(\mathbf{x},t)\,\frac{\partial u(\mathbf{x},t)}{\partial \mathbf{n}} = |_{\Gamma_2}\,\overline{q}(\mathbf{x},t) \tag{4.3}$$

where $\mathbf{x}$ denotes spatial coordinates (x,y), $\mathbf{n}$ is the unit normal vector on the boundary Γ, boundary $\Gamma_1 \cup \Gamma_2 = \Gamma$ and $\Gamma_1 \cap \Gamma_2 = \emptyset$, $u(\mathbf{x},t)$ are unknown displacements on the boundary, $\mathbf{b}$ is the body force, c is the wave speed in the medium, $u_0(\mathbf{x})$ and $v_0(\mathbf{x})$ are given displacements and velocities on the boundary when time $t=0$, $\overline{u}(\mathbf{x},t)$ and $\overline{q}(\mathbf{x},t)$are given displacements and pressure on the boundary Γ_1 and Γ_2 separately at all time.

4.2.2 Green's function and the Boundary Integral Formulation

The boundary integral equation derived in the Chapter 3 also applies to 2D problems. It is written as:

$$\begin{aligned} c\cdot u(\mathbf{x}_0,t) &= \int_0^{t_0^+}\int_S u^*(\mathbf{x},(t-\tau),\mathbf{x}_0)\cdot q(\mathbf{x},\tau)dSd\tau - \int_0^{t_0^+}\int_S q^*(\mathbf{x},(t-\tau),\mathbf{x}_0)\cdot u(\mathbf{x},\tau)dSd\tau \\ &+ \int_0^{t_0^+}\int_V \frac{1}{c^2}(-\mathbf{b}(\mathbf{x},\tau)\cdot u^*(\mathbf{x},(t-\tau),\mathbf{x}_0))dVd\tau \\ &+ \int_0^{t_0^+}\int_V \frac{1}{c^2}\left(\frac{\partial u}{\partial t}(\mathbf{x},0)\cdot u^*(\mathbf{x},(t-\tau),\mathbf{x}_0) + u(\mathbf{x},0)\frac{\partial u^*(\mathbf{x},(t-\tau),\mathbf{x}_0)}{\partial t}\right)dVd\tau \end{aligned} \tag{4.4}$$

where $u(\mathbf{x}_0,t)$ are unknown displacements on the boundary, $u^*(\mathbf{x},t,\mathbf{x}_0)$ and $q^*(\mathbf{x},t,\mathbf{x}_0)$ are Green's functions for displacement and pressure; S is the boundary surface and V is the volume domain the enclosed by the boundary surface. The 2D domain under study can be treated as a special case of a 3D problem where the geometry and boundary conditions are unchanged in the z direction, as shown in Fig. 4.1 .

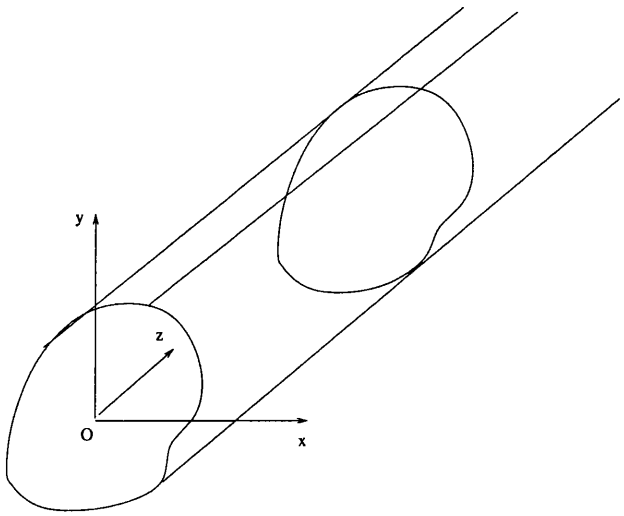


Figure 4.1: The plane and the corresponding 3D prism

The 2D domain Ω and its boundary Γ can be defined as the intersection of the prism with the $x-y$ plane, and $dS = d\Gamma \cdot dz$. The first integral on the right-hand side of Eq. 4.4 can be written as

$$\begin{aligned} & \int_0^{t_0^+} \int_S u^*(\mathbf{x}, t-\tau, \mathbf{x}_0) \cdot q(\mathbf{x}, \tau) dS d\tau \\ = & \int_0^{t_0^+} \int_\Gamma \int_{-\infty}^{\infty} u^*(\mathbf{x}, t-\tau, \mathbf{x}_0) \cdot q(\mathbf{x}, \tau) dz \, d\Gamma d\tau \\ = & \int_0^{t_0^+} \int_\Gamma \int_{-\infty}^{\infty} u^*_{2D}(\mathbf{x}, t-\tau, \mathbf{x}_0) \cdot q(\mathbf{x}, \tau) \, d\Gamma d\tau \end{aligned} \tag{4.5}$$

where the Green's function $u^*_{2D}(\mathbf{x}, t, \mathbf{x}_0)$ is defined by:

$$u^*_{2D}(\mathbf{x}, t-\tau, \mathbf{x}_0) = \int_{-\infty}^{\infty} u^*(\mathbf{x}, t-\tau, \mathbf{x}_0) \, dz \tag{4.6}$$

The same process can be applied for the other integral terms in Eq. 4.4. For the sake of simplicity, assuming zero body sources $\mathbf{b} = \mathbf{0}$ and zero initial condition $u(\mathbf{x}, 0) = 0$ and $\frac{\partial u}{\partial t}(\mathbf{x}, 0) = 0$, the 2D scalar wave BEM equation can be written as:

$$\begin{aligned} c \cdot u(\boldsymbol{x}_0, t) &= \int_\Gamma u^*(\boldsymbol{x}, t-\tau; \boldsymbol{x}_0) \cdot \boldsymbol{q}(\boldsymbol{x}, \tau) d\Gamma d\tau \\ &- \int_\Gamma \boldsymbol{q}^*(\boldsymbol{x}, t-\tau; \boldsymbol{x}_0) \cdot u(\boldsymbol{x}, \tau) d\Gamma d\tau \end{aligned} \tag{4.7}$$

For simplicity, we write $u^*_{2D}(\boldsymbol{x}, t-\tau; \boldsymbol{x}_0) = u^*(\boldsymbol{x}, t-\tau; \boldsymbol{x}_0)$ and $q^*_{2D}(\boldsymbol{x}, t-\tau; \boldsymbol{x}_0) =$

$q^*(\boldsymbol{x}, t-\tau; \boldsymbol{x}_0)$. Then, the Green's functions u^* and q^* are given by (Dominguez [20]):

$$u^*(\boldsymbol{x}, t-\tau, \boldsymbol{x}_i) = \frac{c}{2\pi \left(c^2 t^2 - r^2\right)^{\frac{1}{2}}} H\left(c(t-\tau) - r\right) \tag{4.8}$$

$$q^*(\boldsymbol{x}, t-\tau, \boldsymbol{x}_i) = \frac{\partial r}{\partial n}\left(\frac{cr}{2\pi \left(c^2 (t-\tau)^2 - r^2\right)^{\frac{3}{2}}} H(c(t-\tau) - r) + \frac{c}{2\pi \left(c^2 (t-\tau)^2 - r^2\right)^{\frac{1}{2}}} \frac{\partial H(c(t-\right. \tag{4.9}$$

with $r = |\boldsymbol{x} - \boldsymbol{x}_0|$, $H(x)$ is Heaviside function, and t is the time difference between the source point and field point.

4.2.3 New boundary integral formulation in space-time

Traditionally, dynamic BEM employs a finite difference methodology in time and a boundary element discretization in space. However, in impulse wave problems, regions of high gradients evolve over time and exhibit a high degree of localization at any instant in time. Using smaller time steps for all regions is computationally wasteful. Thus, building a BEM model in which the space-time continuum is discretized in a true sense, combined with an adaptive scheme in both space and time domain should be more efficient. Here we derive a new space-time boundary integral formulation for the general variation in time instead of uniform time stepping.

Putting the explicit form of the Green's functions u^* (Eq. 4.8) and $q^* = \frac{\partial u^*}{\partial n}$ (Eq. 4.9) into Eq. 4.7, the first integral term in the equation Eq. 4.7 is quite straight-forward:

$$\begin{aligned} &\int_\Gamma u^*(\boldsymbol{x}, t-\tau; \boldsymbol{x}_0) \cdot \boldsymbol{q}(\boldsymbol{x}, \tau) d\Gamma d\tau \\ = &\int_\Gamma \frac{c}{2\pi \left(c^2 (t-\tau)^2 - r^2\right)^{\frac{1}{2}}} H\left(ct - r\right) \cdot \boldsymbol{q}(\boldsymbol{x}, \tau) d\Gamma d\tau \end{aligned} \tag{4.10}$$

The second integral term is written as:

$$\int_{\Gamma} q^* u(\boldsymbol{x},\tau)d\Gamma d\tau$$
$$= \int_{\Gamma} \frac{\partial r}{\partial n}\left(\frac{cr}{2\pi\left(c^2(t-\tau)^2-r^2\right)^{\frac{3}{2}}}H(c(t-\tau)-r)+\frac{c}{2\pi\left(c^2(t-\tau)^2-r^2\right)^{\frac{1}{2}}}\frac{\partial H(c(t-\tau)-r)}{\partial r}\right)\cdot u(\boldsymbol{x},\tau)d\Gamma d\tau \tag{4.11}$$

The first term of the right-hand side of Eq. 4.11 is simple. However, in the second term, the derivative of the Heaviside function $\frac{\partial H(c(t-\tau)-r)}{\partial r}$ needs to be treated carefully.

Here the author introduces a new formula for 2D scalar wave propagation with arbitrary shape functions in space-time. Instead of using decoupled shape functions $N(\xi), N(\tau)$, here we use arbitrary shape functions $N^k(\xi,\tau)$ in space-time. Then, *"distance"* in space-time is written as:

$$c(t-\tau)-r = c(t_i - \sum N^k(\xi,\tau)\cdot t_j^k) - r = c(t_i - \Phi(\xi,\tau)) - r \tag{4.12}$$

where $\sum N^k(\xi,\tau)\cdot t_j^k = \Phi(\xi,\tau)$. The derivative of the Heaviside function in Eq. 4.11 is calculated as follows:

$$\frac{\partial H(c(t-\tau)-r)}{\partial r} = \frac{\partial H(c(t_i-\Phi(\xi,\tau))-r)}{\partial r} \tag{4.13}$$

Since it is a Heaviside function, it can be shown (Appendix A):

$$\begin{aligned}\frac{\partial H(c(t-\tau)-r)}{\partial r} &= \frac{1}{c}\cdot\frac{\partial H(c(t-\tau)-r)}{\partial \Phi(\xi,\tau)} \\ &= \frac{1}{c}\cdot\frac{\partial H(c(t-\tau)-r)}{\partial \tau}\frac{\partial \tau}{\partial \Phi(\xi,\tau)} = \frac{1}{c}\cdot\frac{\frac{\partial H(c(t-\tau)-r)}{\partial \tau}}{\frac{\partial \Phi(\xi,\tau)}{\partial \tau}}\end{aligned} \tag{4.14}$$

Substituting Eq. (4.14) back into Eq. (4.11)

$$\int_{\Gamma} q^* u d\Gamma(\boldsymbol{x},t)$$
$$= \int_{\Gamma} \frac{\partial r}{\partial n}\left(\frac{cr}{2\pi\left(c^2(t-\tau)^2-r^2\right)^{\frac{3}{2}}}H(c(t-\tau)-r)+\frac{c}{2\pi\left(c^2(t-\tau)^2-r^2\right)^{\frac{1}{2}}}\frac{1}{c}\cdot\frac{\frac{\partial H(c(t-\tau)-r)}{\partial \tau}}{\frac{\partial \Phi(\xi,\tau)}{\partial \tau}}\right)u(\boldsymbol{x},\tau)\,d\Gamma(\boldsymbol{x},\tau) \tag{4.15}$$

With some rearrangement, the following integral representation for an arbitrary time

shape function is obtained:

$$\begin{aligned} c_i u_i(\boldsymbol{x}_i, t) = & \int_\Gamma u^*(\boldsymbol{x}, t-\tau; \boldsymbol{x}_i) * q(\boldsymbol{x}, \tau) d\Gamma(\boldsymbol{x}, \tau) \\ & - \int_\Gamma \left[z^* u(\xi, \tau) - w^* \frac{\partial u(\xi,\tau)}{\partial \tau} \right] d\Gamma(\boldsymbol{x}, \tau) \end{aligned} \tag{4.16}$$

where z^* and $w^*$are as follows:

$$\begin{aligned} z^* &= \frac{\partial r}{\partial n} \left(\left[\frac{c(r-c(t-\tau))}{2\pi\left(c^2(t-\tau)^2 - r^2\right)^{\frac{3}{2}}} - \frac{\partial}{\partial \tau}\left(\left(\frac{\partial \Phi}{\partial \tau}\right)^{-1}\right) \cdot \frac{1}{2\pi\left(c^2(t-\tau)^2 - r^2\right)^{\frac{1}{2}}} \right] H(c(t-\tau) - r) \right) \\ w^* &= \frac{\partial r}{\partial n} \cdot \left(\frac{\partial \Phi}{\partial \tau}\right)^{-1} \cdot \frac{1}{2\pi\left(c^2(t-\tau)^2 - r^2\right)^{\frac{1}{2}}} \cdot H(ct - r) \end{aligned} \tag{4.17}$$

The author believe that this is the first time that the integral representation of 2D wave propagation for an arbitrary time shape function has ever been derived. This provides a solid mathematical foundation for adaptive schemes in space-time later.

4.3 Discretization of the integral equation for wave 2D

To discretize Eq. 4.16, we divide the geometric boundaries into N_E boundary elements with N nodes, which are associated with N unknown displacements or their normal derivatives. In order to solve those unknown quantities, we write down N equations by applying Eq. 4.16 to each node; this is called the nodal collocation method. Then we have N equations for N unknowns displacements at the nodes. The boundary integral equation is discretized and reduced to a set of linear algebra system equations.

4.3.1 The spatial and temporal discretization

The boundary is divided into linear or quadratic elements in space, and the time axis is divided into equal time steps, which later can be changed into different sizes according to local time stepping. The displacements $u(\mathbf{x}, \tau)$ and pressures $q(\mathbf{x}, \tau)$ are approximated over the boundary elements using the general shape functions in

space-time:

$$u(\mathbf{x},\tau) = \sum_j \sum_m N^{mj}(\xi,\tau)\, u^{mj}$$

$$q(\mathbf{x},\tau) = \sum_j \sum_m N^{mj}(\xi,\tau)\, q^{mj} \tag{4.18}$$

where $N^{mj}(\xi,\tau)$ is a two-dimensional shape function in space-time. The superscripts j and m denote the point j at the time step m; the superscript k denotes the kth element; and u, q represents the displacements and the pressures, respectively. After the numerical approximation, Eq. (4.16) takes the discrete form:

$$\begin{aligned} c_i u_i^n &= \sum_{m=1}^{N}\sum_{k=1}^{K}\int\int_{\Gamma_k} u^* \cdot N^{mj}(\xi,\tau)\, q^{mj} d\Gamma d\tau \\ &- \int\int_{\Gamma_k}\left[z^* N^{mj}(\xi,\tau) - w^*\frac{\partial N^{mj}(\xi,\tau)}{\partial\tau}\right]\cdot u^{mj} d\Gamma d\tau \end{aligned} \tag{4.19}$$

where Γ_k indicates the kth boundary element, u^{mj} and q^{mj} are the nodal values of the displacement and the pressure at the point j at the time step m; N is the total number of time steps, and K is the total number of elements.

Letting

$$\begin{aligned} G_{ij}^{nm} &= \int\int_{\Gamma_j} u^* \cdot N^{mj}(\xi,\tau)\, d\Gamma d\tau \\ \hat{H}_{ij}^{nm} &= \int\int_{\Gamma_j}\left[z^* N^{mj}(\xi,\tau) - w^*\frac{\partial N^{mj}(\xi,\tau)}{\partial\tau}\right]\cdot d\Gamma d\tau \end{aligned} \tag{4.20}$$

Then Eq. 4.24 can be written as

$$c_i u_i^n = \sum_{m=1}^{N}\sum_{j=1}^{J}[G_{ij}^{nm} q^{mj} - \hat{H}_{ij}^{nm} u^{mj}] \tag{4.21}$$

Adopting the notation:

$$H_{ij}^{nm} = \begin{cases} \hat{H}_{ij}^{nm} & i \neq j \\ \hat{H}_{ij}^{nm} + c_i & i = j \end{cases} \tag{4.22}$$

the system of equations for all boundary nodes can be expressed as:

$$\sum_{m=1}^{N}\sum_{j=1}^{J} H_{ij}^{nm} u_j^m = \sum_{m=1}^{N}\sum_{j=1}^{J} G_{ij}^{nm} q_j^m \tag{4.23}$$

or in matrix form:

$$\sum_{m=1}^{N} \mathbf{H}^{nm}\mathbf{u}^m = \sum_{m=1}^{N} \mathbf{G}^{nm}\mathbf{q}^m \tag{4.24}$$

where $\mathbf{H}^{nm}$ and $\mathbf{G}^{nm}$ are matrices whose elements are obtained by integration over boundary elements in the time interval m while the collocation point is in the time interval n.

Eq. 4.24 is solved step-by-step. Once $\mathbf{u}^m$ and $\mathbf{q}^m$ are known for previous time steps, the solution for time step n is obtained from

$$\mathbf{H}^{nn}\mathbf{u}^n = \mathbf{G}^{nn}\mathbf{q}^n + \sum_{m=1}^{n-1} \mathbf{G}^{nm}\mathbf{q}^m - \mathbf{H}^{nm}\mathbf{u}^m \tag{4.25}$$

The columns of $\mathbf{H}^{nm}$ and $\mathbf{G}^{nm}$ can be swapped to separate known boundary conditions (sent to the right-hand side) and unknown ones (kept in the left-hand side). Eq. 4.24 becomes a square linear system of equations:

$$\mathbf{K}^n\mathbf{X}^n = \mathbf{F}^n \tag{4.26}$$

where the known vector $\mathbf{F}^n$ contains the second term on the right-hand side of Eq. 4.25, which corresponds to all previous influences, combined with the product of the right-hand side matrix (re-ordered) and the known vector of the boundary conditions at time step n.

4.3.1.1 Solution of BEM equations

For 2D wave problems, the matrix $\mathbf{K}^n$ is fully populated and non-symmetric. Eq. 4.26 is usually solved by Gauss Elimination (an $O(N^3)$ procedure) or the General Minimal Residual iterative algorithm (GMRES) (E. Stein [47]). Given the computational demands of this approach, an adaptive scheme (Section 4.4) for the impulse

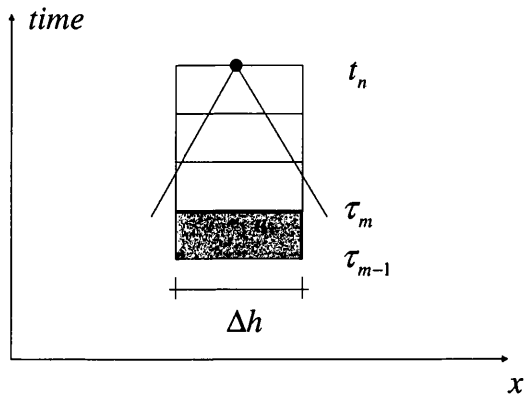


Figure 4.2: All signals from τ_{m-1} and τ_m has arrived

wave problem appears to be profitable.

4.3.2 Evaluation of time integral

Generally in the space-time scheme, the time integral is treated by using numerical quadrature, like the spatial integrals. However, since the adaptive scheme will be introduced on the top of a uniform mesh, the time integral on the uniform mesh is evaluated analytically to maintain efficiency and accuracy. We follow Dominguez [20] for the analytical evaluation of the time integral. Firstly, we split the space-time shape function $N^{mj}(\xi,\tau)$ into the time shape function $\mu^m(\tau)$ and the space shape function $\phi(\xi)$. To evaluate the time integral of Eq. 4.19, we assume a piece-wise linear approximation for u and a piece-wise constant one for $\frac{\partial u}{\partial n}$, that is, $\mu^m(\tau)=1$ for $\tau_{m-1}<\tau<\tau_m$ and $\mu^m(\tau)=0$ elsewhere. The first term of the time integral is:

$$U^{nm}=\int_0^{t^+} u^*\mu^m(\tau)d\tau=\int_{\tau_{m-1}}^{\tau_m^+}\frac{cH[c(t_n-\tau)-r)}{2\pi\sqrt{c^2(t_n-\tau)^2-r^2}}d\tau \tag{4.27}$$

Eq. 4.27 is the integral of the effect on point $\mathbf{x}^i$ at t_n from a source point at $\mathbf{x}^j$ between τ_{m-1} and τ_m. There are three cases to consider:

1) $r<c(t_n-\tau_m)$

All of the signal has arrived, as shown in Fig. 4.2.

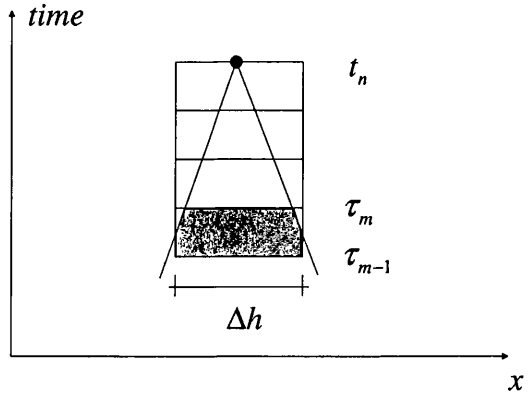


Figure 4.3: Part of signal from τ_{m-1} and τ_m has arrived

$$\begin{aligned} U^{nm} &= \int_{\tau_{m-1}}^{\tau_m^+} \frac{c}{2\pi\sqrt{c^2(t_n-\tau)^2-r^2}} d\tau \\ &= \frac{1}{2\pi} ln \frac{\alpha_0+\sqrt{\alpha_0^2-1}}{\alpha_1+\sqrt{\alpha_1^2-1}} \end{aligned} \tag{4.28}$$

where

$$\alpha_0 = \frac{c(t_n-\tau_{m-1})}{r} \qquad and \qquad \alpha_1 = \frac{c(t_n-\tau_m)}{r}$$

2) $c(t_n-\tau_m) \le r < c(t_n-\tau_{m-1})$

Part of the signal has arrived, as shown in Fig. 4.3.

$$\begin{aligned} U^{nm} &= \int_{\tau_{m-1}}^{\tau=t_n-\frac{r}{c}} \frac{c}{2\pi\sqrt{c^2(t_n-\tau)^2-r^2}} d\tau \\ &= \frac{1}{2\pi} ln \left[\alpha_0+\sqrt{\alpha_0^2-1}\right] \end{aligned} \tag{4.29}$$

3) $r \ge c(t_n-\tau_{m-1})$

None of signal has arrived, as shown in Fig. 4.4.

$$U^{nm} = 0 \tag{4.30}$$

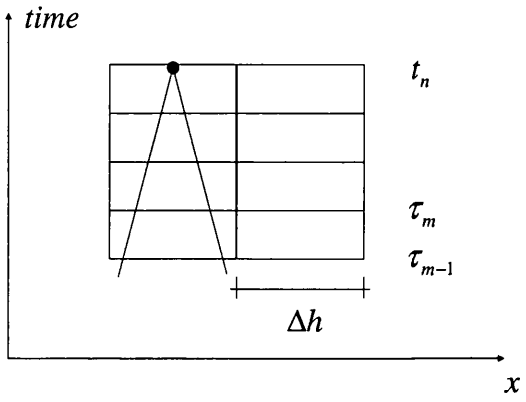


Figure 4.4: No signal from τ_{m-1} and τ_m has arrived

The singularity only appears when $t_n = \tau_m$ and $r \to 0$. This singularity is of the same type as that in the static problem. All other space integrals can be done by using standard Gauss quadrature, whereas for $t_n = \tau_m$ and $r \to 0$, analytical integration is necessary.

For the uniform space-time mesh, we assume the time shape function $\mu^m(\tau)$ for the displacement u to be linear. The second term of the time integral (Eq. 4.16) is:

$$I_2 = \int_\Gamma \left[z^* u\left(\xi,\tau\right) - w^* \frac{\partial u\left(\xi,\tau\right)}{\partial \tau} \right] d\Gamma d\tau \tag{4.31}$$

Substituting the numerical approximation of the displacement u:

$$\begin{aligned} I_2 &= \int_\Gamma \left[z^* u\left(\xi,\tau\right) - w^* \frac{\partial u\left(\xi,\tau\right)}{\partial \tau} \right] d\Gamma d\tau && (4.32) \\ &= \int_\Gamma \left[z^{*m}(\tau) \cdot u^m\left(\xi,\tau\right) - w^* \frac{\partial^m(\tau) \cdot u^m\left(\xi,\tau\right)}{\partial \tau} \right] d\Gamma d\tau && (4.33) \\ &= \int_\Gamma {}^m\left(\xi,\tau\right) d\Gamma && (4.34) \end{aligned}$$

where

$$Q^{nm} = \int_\Gamma \left[z^* \mu^m(t^n - \tau) - w^* \frac{\partial \mu^m(t^n - \tau)}{\partial \tau} \right] d\tau \tag{4.35}$$

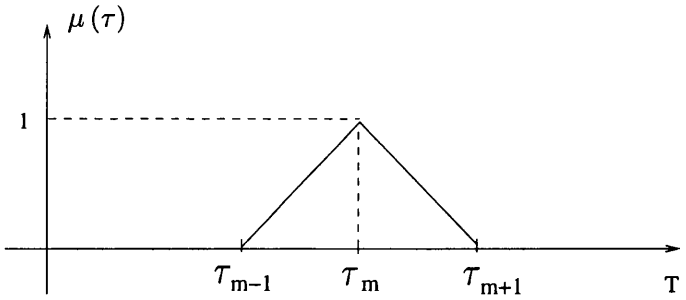


Figure 4.5: Linear time shape function for u

where z^* and w^* are given by Eq. 4.17, t^n is the current time and n is current time step, m are the past time steps .

The shape function $\mu^m(\tau)$ varies linearly from 0 to 1, as shown in Fig. 4.5.

$$\mu^m(\tau) = \begin{cases} \frac{1}{\triangle t}(\tau - \tau_{m-1}) & \tau_{m-1} \leq \tau \leq \tau_m \\ \frac{1}{\triangle t}(\tau_{m+1} - \tau) & \tau_m \leq \tau \leq \tau_{m+1} \end{cases} \tag{4.36}$$

Substituting Eq. 4.17 and Eq. 4.36 into Eq. 4.35, yields:

$$\begin{aligned} Q^{nm} &= \int_{\tau_{m-1}}^{\tau_m^+} \frac{\partial r}{\partial n} \frac{(\tau - \tau_{m-1})}{\triangle t} \frac{c[r - c(t-\tau)]}{2\pi(c^2(t_n - \tau)^2 - r^2)^{\frac{3}{2}}} H[r - c(t-\tau)] d\tau \\ &- \int_{\tau_{m-1}}^{\tau_m^+} \frac{\partial r}{\partial n} \frac{1}{\triangle t} \frac{1}{2\pi(c^2(t_n - \tau)^2 - r^2)^{\frac{1}{2}}} H[r - c(t-\tau)] d\tau \\ &+ \int_{\tau_m}^{\tau_{m+1}^+} \frac{\partial r}{\partial n} \frac{(\tau_{m+1} - \tau)}{\triangle t} \frac{c[r - c(t-\tau)]}{2\pi(c^2(t_n - \tau)^2 - r^2)^{\frac{3}{2}}} H[r - c(t-\tau)] d\tau \\ &- \int_{\tau_m}^{\tau_{m+1}^+} \frac{\partial r}{\partial n} \frac{-1}{\triangle t} \frac{1}{2\pi(c^2(t_n - \tau)^2 - r^2)^{\frac{1}{2}}} H[r - c(t-\tau)] d\tau \end{aligned} \tag{4.37}$$

This integral can be considered in four cases:

1) $r < c(t_n - \tau_{m+1})$

All of the signal has arrived:

$$Q^{nm} = \frac{-\partial r}{\partial n} \frac{1}{2\pi c \triangle t} (\sqrt{\alpha_0^2 - 1} - \sqrt{\alpha_1^2 - 1} + \sqrt{\alpha_2^2 - 1}) \tag{4.38}$$

where

$$\alpha_0 = \frac{c(t_n - \tau_{m-1})}{r} \quad and \quad \alpha_1 = \frac{c(t_n - \tau_m)}{r} and \quad \alpha_2 = \frac{c(t_n - \tau_{m+1})}{r}$$

2) $c(t_n - \tau_{m+1}) \leq r < c(t_n - \tau_m)$

Part of the signal has arrived:

$$Q^{nm} = \frac{-\partial r}{\partial n} \frac{1}{2\pi c \triangle t} (\sqrt{\alpha_0^2 - 1} - \sqrt{\alpha_1^2 - 1}) \tag{4.39}$$

3) $c(t_n - \tau_m) \leq r < c(t_n - \tau_{m-1})$

Part of the signal has arrived:

$$Q^{nm} = \frac{-\partial r}{\partial n} \frac{1}{2\pi c \triangle t} (\sqrt{\alpha_0^2 - 1}) \tag{4.40}$$

4) $r \geq c(t_n - \tau_{m-1})$

None of the signal has arrived:

$$Q^{nm} = 0 \tag{4.41}$$

The singularity only appears when $t_n = \tau_m$ and $r \rightarrow 0$. This singularity is of the same type as that in the static problem. All other space integrals can be done by using standard Gauss quadrature, whereas for $t_n = \tau_m$ and $r \rightarrow 0$, analytical integration is necessary.

Once the coefficients of U^{nm} and Q^{nm} have been calculated, the entries of G_{ij}^{nm} and H_{ij}^{nm} can be obtained by space integration along the boundary elements, as described in the next section.

4.3.3 Numerical schemes for space integrals

4.3.3.1 Evaluation of non-singular integrals

The space integrations along the boundary elements, except the singular case $t_n = \tau_m$ and $r \rightarrow 0$, can be done by standard Gauss quadrature after introducing the space shape function ϕ_j, which is defined as follows:

$$\phi_1(\xi) = \frac{\xi(\xi-1)}{2} \qquad \phi_2(\xi) = (1+\xi)(1-\xi) \qquad \phi_3(\xi) = \frac{\xi(\xi+1)}{2} \tag{4.42}$$

The isoparametric numerical approximation for physical quantities is used here:

$$\begin{aligned} x &= \phi_1(\xi)\, x_1 + \phi_2(\xi)\, x_2 + \phi_3(\xi)\, x_3 \\ y &= \phi_1(\xi)\, y_1 + \phi_2(\xi)\, y_2 + \phi_3(\xi)\, y_3 \\ u(\xi) &= \phi_1(\xi)\, u_1 + \phi_2(\xi)\, u_2 + \phi_3(\xi)\, u_3 \\ q(\xi) &= \phi_1(\xi)\, q_1 + \phi_2(\xi)\, q_2 + \phi_3(\xi)\, q_3 \end{aligned} \tag{4.43}$$

Substituting time integrals Eq. 4.27 and Eq. 4.35 into the discrete integral representation Eq. 4.19, it becomes:

$$c_i u_i^n = \sum_{m=1}^{N} \sum_{j=1}^{J} \int_{\Gamma_j} U^{nm}\left(x^i, t-\tau, x^j(\xi)\right) \phi_j q_j^m d\Gamma(\mathbf{x}) - \int_{\Gamma_j} Q^{nm}\left(x^i, t-\tau, x^j(\xi)\right) \cdot \phi_j u^{mj} d\Gamma(\mathbf{x}) \tag{4.44}$$

where $U^{nm}\left(x^i, t-\tau, x^j(\xi)\right)$ and $Q^{nm}\left(x^i, t-\tau, x^j(\xi)\right)$ are analytical integrals of the time kernel, u^{mj} and q^{mj} are displacements and pressures at element j at time step m. The total number of elements is J, and total number of time steps is N.

Substituting the quadratic shape functions of Eq. 4.42 into Eq. 4.44, we obtain:

$$\int_{\Gamma_j} U^{nm} \phi_j q_j^m d\Gamma(\mathbf{x}) = \int_{\Gamma_j} U^{nm}\left(x^i, t-\tau, x^j(\xi)\right) [\phi_1\ \phi_2\ \phi_3] d\Gamma_j(\boldsymbol{x}) \begin{Bmatrix} q_{j1}^m \\ q_{j2}^m \\ q_{j3}^m \end{Bmatrix}$$

$$\int_{\Gamma_j} Q^{nm} \cdot \phi_j u^{mj} d\Gamma(\mathbf{x}) = \int_{\Gamma_j} Q^{nm}\left(x^i, t-\tau, x^j(\xi)\right) [\phi_1\ \phi_2\ \phi_3] d\Gamma_j(\boldsymbol{x}) \begin{Bmatrix} u_{j1}^m \\ u_{j2}^m \\ u_{j3}^m \end{Bmatrix} \tag{4.45}$$

After applying Gauss quadrature, the above integrals are written as:

$$G_{ij}^{nm} = \sum_{\alpha=1}^{Ng} q_{j\alpha}^{m} \cdot U^{nm}\left(x^i, t-\tau, x^j(\xi)\right) \phi_\alpha(\xi) J(\xi) \cdot w_\alpha$$

$$\hat{H}_{ij}^{nm} = \sum_{\alpha=1}^{Ng} u_{j\alpha}^{m} Q^{nm}\left(x^i, t-\tau, x^j(\xi)\right) \phi_\alpha(\xi) J(\xi) \cdot w_\alpha \tag{4.46}$$

where w_α is the weight of the Gauss quadrature, N_g is the total number of the Gauss points. $\phi_\alpha(\xi)$ are the shape functions defined in Eq. 4.42, U^{nm}, Q^{nm} are time integral kernel functions, $u_{j\alpha}^{m}, q_{j\alpha}^{m}$ are the displacements and their normal derivatives on the boundary. $J(\xi)$ is the Jacobian, which is defined as follows:

$$J(\xi) = \sqrt{\left(\frac{dx}{d\xi}\right)^2 + \left(\frac{dy}{d\xi}\right)^2} \tag{4.47}$$

can be calculated from Eq. 4.42.

Owning to the $O(lnr)$ or the $O(r^{-1})$ behaviour of the Green's functions, it is more efficient to adjust the number of Gauss points used according to the distance between $\mathbf{x}^i$ and $\mathbf{x}^j$. More details can be found in (X. W. Gao and T. G. Davies [27]).

4.3.3.2 Evaluation of the singular integrals

If the integral is evaluated at the first time step and the collocation point is in the integration element, the Green's function u^* has a singularity $O(ln\, r)$ while q^* has a singularity which can be removed according to the following arguments:

$$q^* = -\frac{\partial r}{\partial n} \frac{1}{2\pi \triangle t} \frac{(c^2(t_n-\tau)^2 - r^2)^{\frac{1}{2}}}{r} \tag{4.48}$$

The denominator goes to zero as $r \rightarrow 0$, but $\frac{\partial r}{\partial n}$ goes to zero with order $O(r)$ as well. Thus, $q^* = O(1)$ as $r \rightarrow 0$. The integral of the normal derivative can be done through standard Gauss quadrature as described in the previous section.

However, the singular integrals of the Green's function u^* have to be evaluated by a special numerical scheme. The time integral of Green's function u^* is given by Eq. 4.29. At the first time step, when $t_n = \triangle t$ and $\tau_{m-1} = 0$, it becomes:

$$U^{nm} = \frac{1}{2\pi} ln \left(c \triangle t + \left(c^2 \triangle t^2 - r^2 \right)^{\frac{1}{2}} \right) + \frac{1}{2\pi} ln \frac{1}{r} \tag{4.49}$$

Substituting the above into the first term of the right-hand side of Eq. 4.16:

$$\int_{\Gamma_j} u^* q d\Gamma_j d\tau = \frac{1}{2\pi} \int_{\Gamma_j} \left[ln \left(c \triangle t + \left(c^2 \triangle t^2 - r^2 \right)^{\frac{1}{2}} \right) + ln \frac{1}{r} \right] q d\Gamma_j \tag{4.50}$$

The second term of the above equation has a logarithmic singularity when $r \rightarrow 0$. We can separate the logarithmic singularity and integrate it using the Gauss integration rule for logarithmically singular functions (The nonsingular terms can be easily integrated using ordinary Gauss quadrature). The Gauss quadrature rule for a logarithmic function takes the form:

$$I = \int_0^1 log_e \frac{1}{x} f(x) dx \cong \sum_{k=1}^{n} w_k f(x_k) \tag{4.51}$$

where the interval of integration is from zero to unity and the ordinates x_k and the weights w_k can be found in (X. W. Gao and T. G. Davies [27]).

When all entries of influence matrices G_{ij} and H_{ij} are obtained, they are assembled into matrices $[G]$ and $[H]$. Once matrices $[G]$ and $[H]$ have been obtained, the matrix equation can be solved using standard matrix solvers as described in Section 4.3.1.1.

4.4 Error indicators and adaptive scheme

In this section, we establish error representation formulas and show how the boundary element solution error can be estimated *a posteriori* , that is, with knowledge based on postprocessing the BEM solution.

4.4.1 Hierarchical type error estimation in 2D wave propagation

The basic idea of hierarchical type error estimation is to compute an approximation $\tilde{u}$ with hierarchically expanded shape functions at a low computational cost. Supposing the discrete space where u_h lies in is V_h, the hierarchical expanded space $\tilde{V}$, total solution space is V. The hierarchically refined solution $\tilde{u}$ is a better approximation of the exact solution $u \in V$ than the former discrete solution $u_h \in V_h$ according to the Saturation Assumption (Hsiao & Wendland [35]):

$$\|u - \tilde{u}\| \leq C \|u - u_h\| \tag{4.52}$$

where $\|\cdot\|$ is an energy norm, C is a constant $C \in [0, 1)$. It means that we assume that $\tilde{u} \in \tilde{V}$ converges faster to $u \in V$ than $u_h \in V_h$. With the saturation assumption and the triangle inequality we can derive a posterior error estimator:

$$\frac{1}{1+C} \|\tilde{u} - u_h\| \leq \|e_u\| \leq \frac{1}{1-C} \|\tilde{u} - u_h\| \tag{4.53}$$

where e_u are point-wise errors of displacement.

The point-wise errors of normal derivatives e_q satisfy the same condition:

$$\frac{1}{1+C} \|\tilde{q} - q_h\| \leq \|e_q\| \leq \frac{1}{1-C} \|\tilde{q} - q_h\| \tag{4.54}$$

Theoretically, point-wise errors are defined as follows:

$$\begin{aligned} e_u &= u - u_h = \delta u \\ e_q &= q - q_h = \delta q \end{aligned} \tag{4.55}$$

where e_u and e_q are the point-wise errors of displacements and tractions, u, q are exact values which satisfy the governing equation, while u^h, q^h are values in the discrete space of the same problem. Based on point-wise errors , we can derive element-wise errors in L_2 norm, which dictate where to refine the boundary element

mesh further.

$$\|e_u\| = \left[\int \left(\sum_{i=1}^{K} N_i \delta u_i\right)^2 d\Gamma\right]^{\frac{1}{2}} \qquad \|e_q\| = \left[\int \left(\sum_{i=1}^{K} N_i \delta q_i\right)^2 d\Gamma\right]^{\frac{1}{2}} \tag{4.56}$$

where N_i are shape functions used in the space-time domain, δu_i and δq_i are point-wise errors of displacements and normal derivatives in the mesh, which are obtained from Eq. (4.55), and K is the total number of nodes within one element. The central idea is to find the way to compute $\tilde{u} \in \tilde{V}$ cheaply after we obtain the boundary element solution $u_h \in V_h$. For example, the results obtained with coarse and refined mesh, taken as $\tilde{u}$ and u_h, can be used to calculate point-wise errors. The advantage of this strategy is that we combine two processes of error estimation and adaptivity by continuing the refinement of elements whose errors are big while stopping the refinement of elements whose errors are small. The adaptive process will be described in more detail in the next section.

4.4.2 Error indicator and adaptive scheme

Given *a posteriori* global error estimator, we now use a h- adaptive scheme to refine the current BEM mesh, as described in Sec. 4.4.2.1. The strategy to select elements for refinement is described in Section 4.4.2.2.

4.4.2.1 h- adaptive scheme

For the BEM 2D problem, curved line elements are usually refined by halving the marked elements. However, it is inefficient for the whole influence matrix to be recalculated after each refinement. Instead, a hierarchical h- adaptive scheme is employed here:

$$u(x,\tau) = N^m(\xi,\tau)\, u^m + N^\alpha(\xi,\tau)\, u^\alpha \tag{4.57}$$

where $N^m(\xi,\tau)$ and u^m are the ordinary shape functions and displacements on the root mesh; $N^\alpha(\xi,\tau)$ and u^α are the hierarchical shape functions and extra freedoms associated with them. The simplest hierarchical orthogonal shape functions

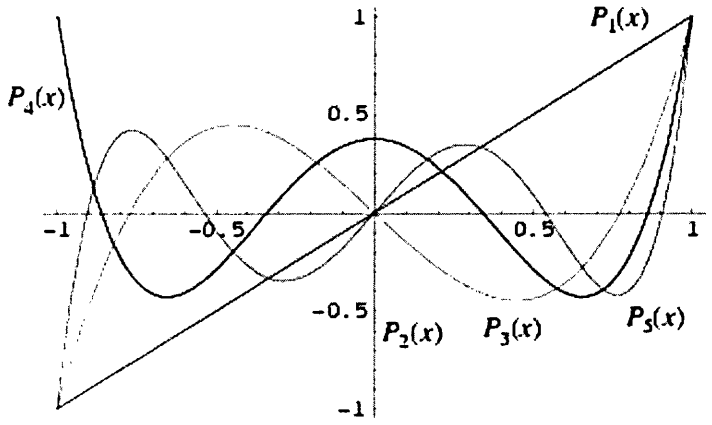


Figure 4.6: Legendre polynomials as hierarchical shape functions

$N^{\alpha}(\xi,\tau)$ can be based on Legendre polynomials or on standard trigonometric functions (Robert L.Taylor and O.C.Zienkiewicz and E.Oñate [49]), as shown in Fig. 4.6. More advanced hierarchical shape functions based on the concept of the partition of unity can also be used (Melenk, [41]).

4.4.2.2 Adaptive process

When the error estimates for each element are known, the global error over the whole mesh can be simply obtained by summing them up.

$$\eta^{u} = \sum_{i=1}^{N} e_{ui} \qquad \eta^{q} = \sum_{i=1}^{N} e_{qi} \tag{4.58}$$

where N is the total number of elements.

The adaptive process for the solution of 2D wave propagation is:

Step 1: Apply the coarse mesh on the boundary of the problem domain;

Step 2: Carry out the BEM analysis;

Step 3: Apply the refinement to all elements at the first time step. Compute local node errors δu and δq; the element errors e_u, e_q and the global errors of η^{u}, η^{q}. Then, locate the maximum of the local errors:

$$\eta^{u}_{\max} = Max(e_u) \ , \qquad \eta^{q}_{\max} = Max(\ e_q) \tag{4.59}$$

Step 4: An element is refined if the element error indicators e_u, e_q are bigger

than $tol.\eta^u_{\max}$, $tol.\eta^q_{\max}$, where tol is an error tolerance given beforehand, which is usually chosen in the range 0.5 and 0.8.

$$e_u > tol.\eta^u_{\max} \qquad e_q > tol.\eta^q_{\max} \tag{4.60}$$

We stop the adaptive process if the following conditions are satisfied:

$$\eta^u < \bar{\eta}^u_G \quad and\ \eta^q < \bar{\eta}^q_G \tag{4.61}$$

here $\bar{\eta}^u_G$ and $\bar{\eta}^q_G$ are the principal global error tolerances. Since absolute errors are unknown, the stop condition for the adaptive scheme depends on experience and accuracy required. Here the error tolerance is chosen around $3\% \sim 5\%$. (Marcus Rüter and Erwin Stein, [73])

Step 5. Use the hierarchical adaptive scheme to enrich the approximation for those elements in space, use half time step at the marked time steps for all elements, then back to step 2.

The adaptive process is shown in Fig. 4.7.

4.5 Numerical Examples

Some examples are presented in this section to demonstrate the performance of space-time BEM to improve accuracy and stability of the 2D wave equation.

4.5.1 2D Rod

The numerical example here is the simple plane wave problem where a strip of height $L = 8m$, extending indefinitely in the y direction, has an uniform pressure $P(t)$ applied on the upper surface, where $P(t)$ is the Heaviside function. The wave speed is $200m/s$. This is shown in Fig. 4.8.

This problem is essentially one-dimensional in space, and has been widely used as benchmark since the jump load causes instabilities in numerical methods. Also, the analytical solution is available. Here we solve it in two dimensional space-time by taking a strip of $8m \times 4m$ and meshing it in the full space-time domain. The

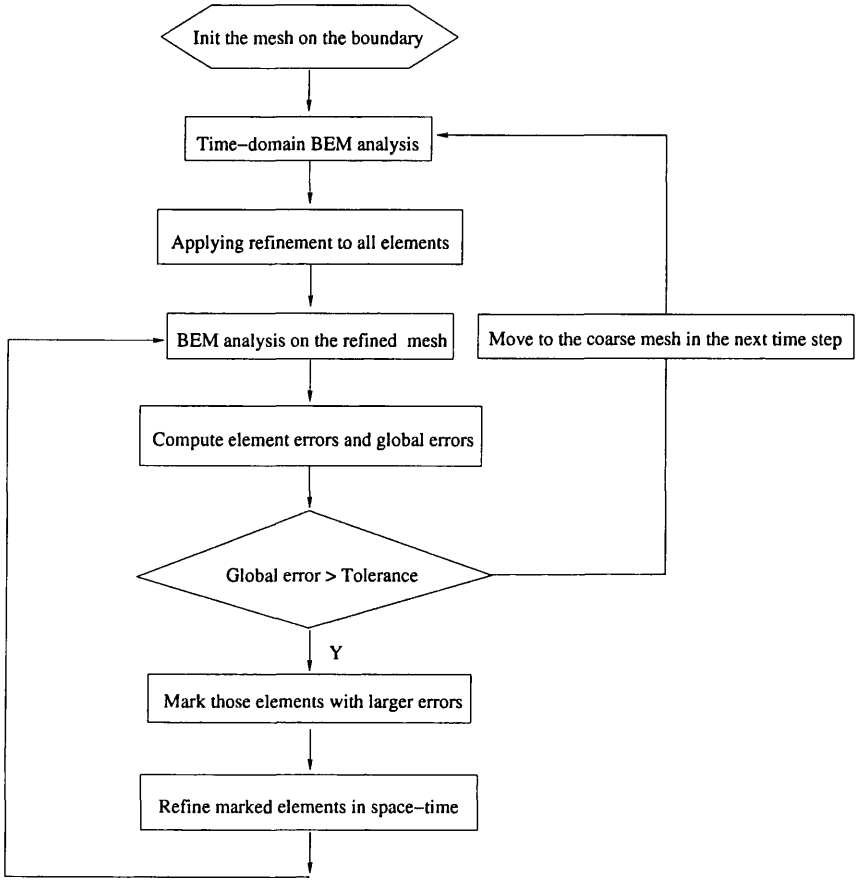


Figure 4.7: The flowchart of adaptive scheme

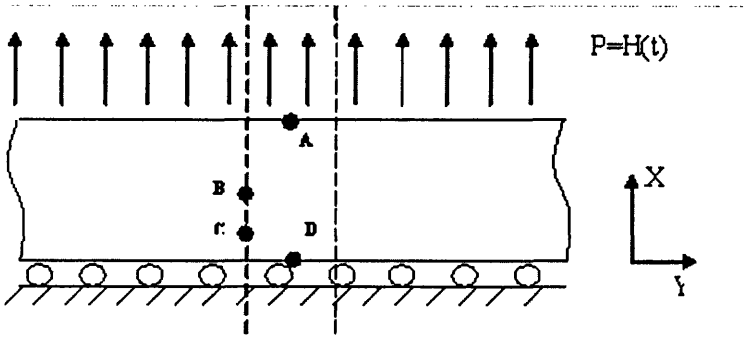


Figure 4.8: Plane wave problem.

boundary was initially discretized into 600 constant and linear mixed elements in space-time. The space-time ratio $\beta = 1$; the mesh size is $4m$; the time step is $0.02s$, and there are 100 time steps. We initiate the adaptive scheme by refining the whole mesh. The process is schematically illustrated in Fig. 4.9.

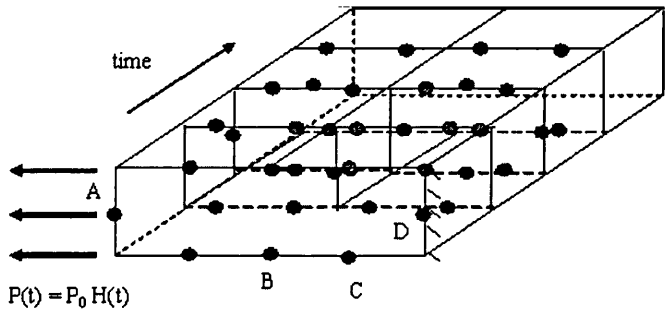


Figure 4.9: h- adaptive scheme in space-time 2D.

First we presents some results to demonstrate the numerical difficulties in this problem: small time steps lead to unstable results, while big time steps lead to significant numerical damping and loss of accuracy, as shown in Fig. 4.10. The most important parameter to define the relative time step size to its mesh size is space-time ratio β:

$$\beta = \frac{c \cdot \Delta t}{\Delta h} \tag{4.62}$$

When $\beta = 0.1$, the result is obviously unstable; When $\beta = 0.3$, the result improves but is still unstable; When $\beta = 0.5$, the results becomes unstable again; when $0.5 < \beta < 1.0$, and especially when $\beta = 0.7$, the solution is optimum for stability and accuracy. When $\beta > 1.2$, the results gradually lose accuracy but gain stability. Similar results can be found for the structure under an impulse load (Fig. 4.11), as shown in Fig. 4.12. According to these results, accuracy and stability appears to be mutually exclusive of conventional dynamic BEM.

Now we consider the adaptive BEM scheme for this problem. After refining the mesh to 12 elements (in space), we draw the graph of relative element errors e_i^u,e_i^q in Fig. 4.13. The X axis represents the element number, the Y axis represents the time step, and the Z axis represents relative element errors. Since the upper side of the rectangle has the same solution as the lower one, displacement errors of

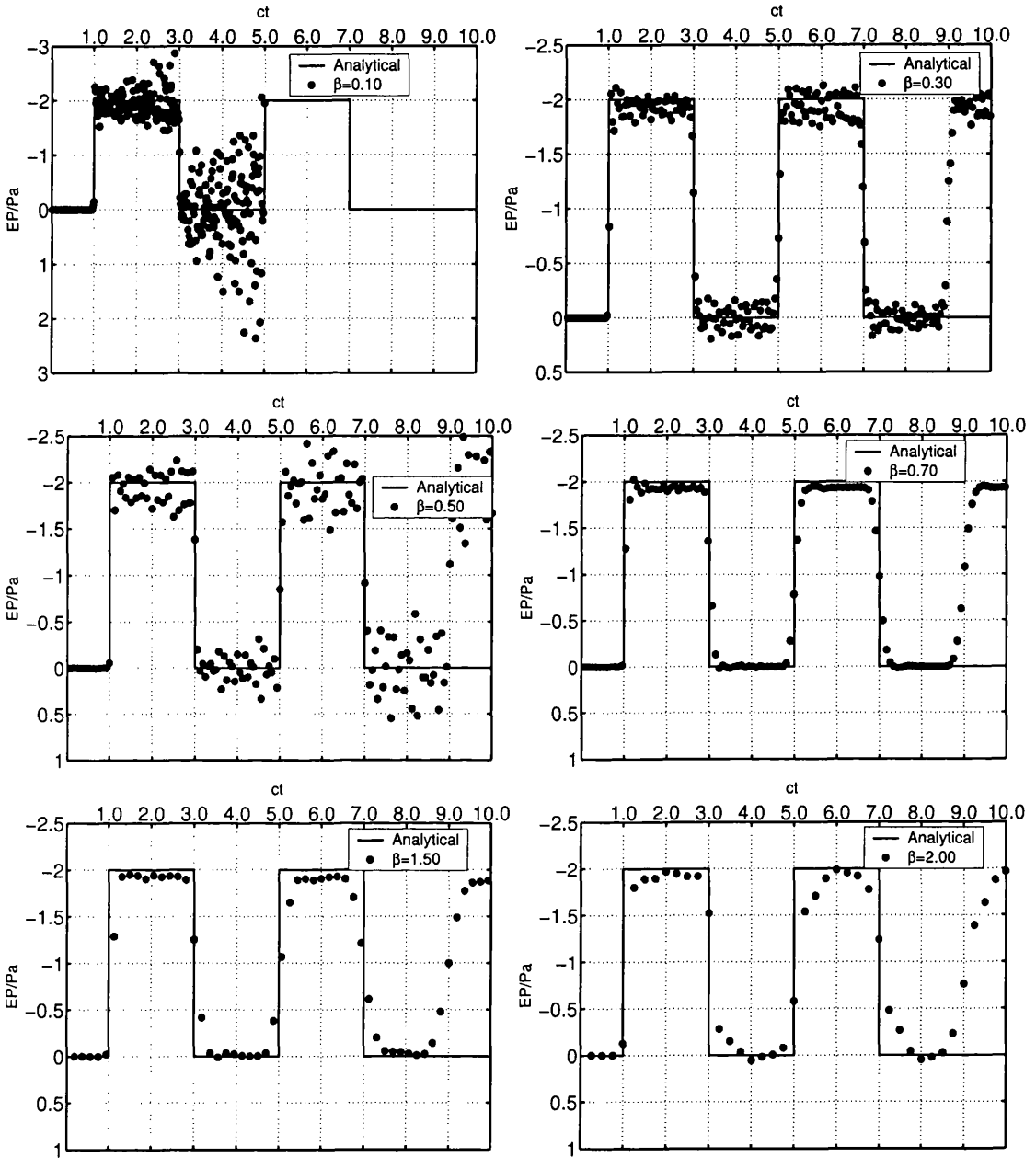


Figure 4.10: Time history of the normal traction at the fixed end (point D) with different β, under a Heaviside Load

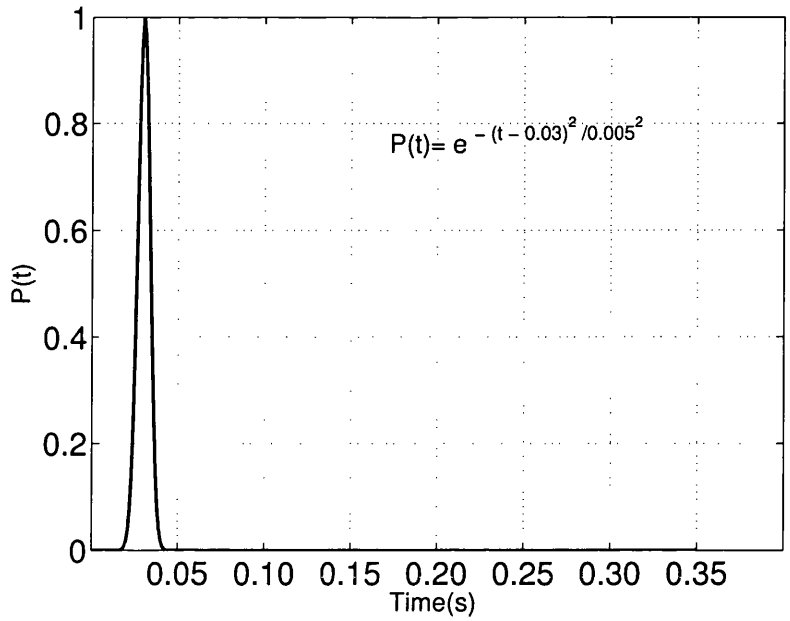


Figure 4.11: Gaussian type impulse load

elements $1-6$ are symmetric to those of elements $7-12$. Here we can see that the error estimates indicate the correct position of the moving wave front in the graph of displacement errors. Because q are zeroes at most places except at the fixed end, there is only one narrow band in the graph of pressure errors. It has peak values when the wave front reach the fixed end and drops to zero when q remains constant afterward. The time history of q at the fixed end can be seen in Fig. 4.14.

After the solution errors are located in the BEM mesh, we use hierarchical refinement and smaller time steps near the wave front and re-compute it. Improvements near the area of the wave front at points A, B, C, D are observed in Fig. 4.14. Without the refinement scheme, the solution near the rectangular wave fronts is seriously damped while the refined one offers a better approximation.

4.5.2 Transient loads on the surface of half-plane

We consider the half-plane under a surface load uniformly distributed over a length $2b = 152m$, the wave speed in the media is $c = 200m/s$. The load is suddenly applied and maintained there, as shown in Fig. 4.15. (Manolis [51]). The dynamic response at point A, which is at the center of the applied load, is sought after the Heaviside function is applied. This is a benchmark which is widely used for BEM in

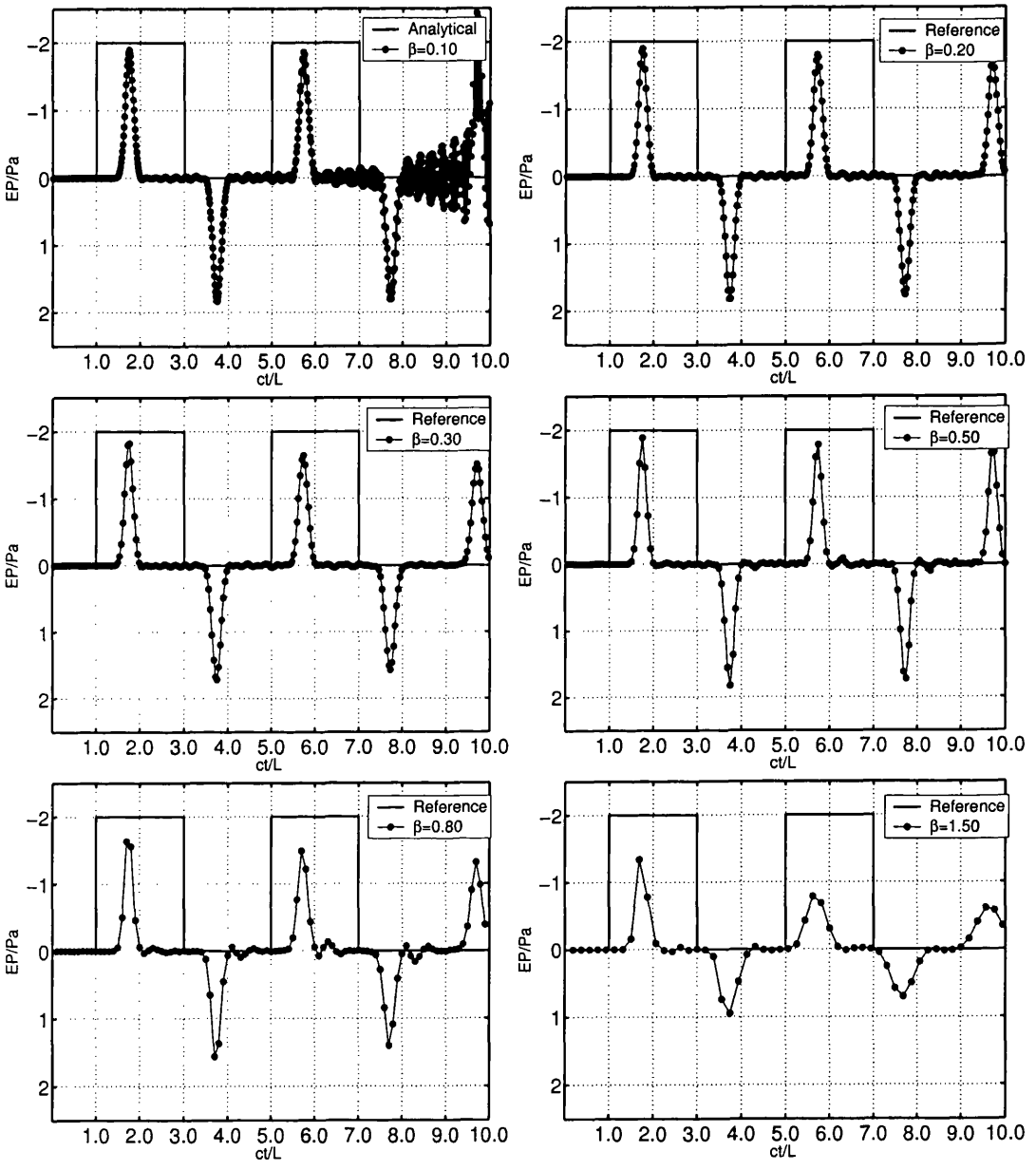


Figure 4.12: Time history of the normal traction at the fixed end (point D) with different β, under an impulse load

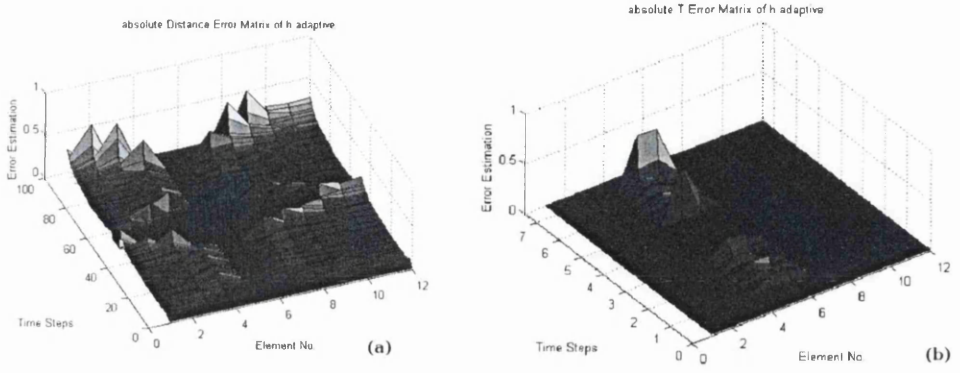


Figure 4.13: (a) Relative element error of displacement; (b) Relative element errors of normal derivatives

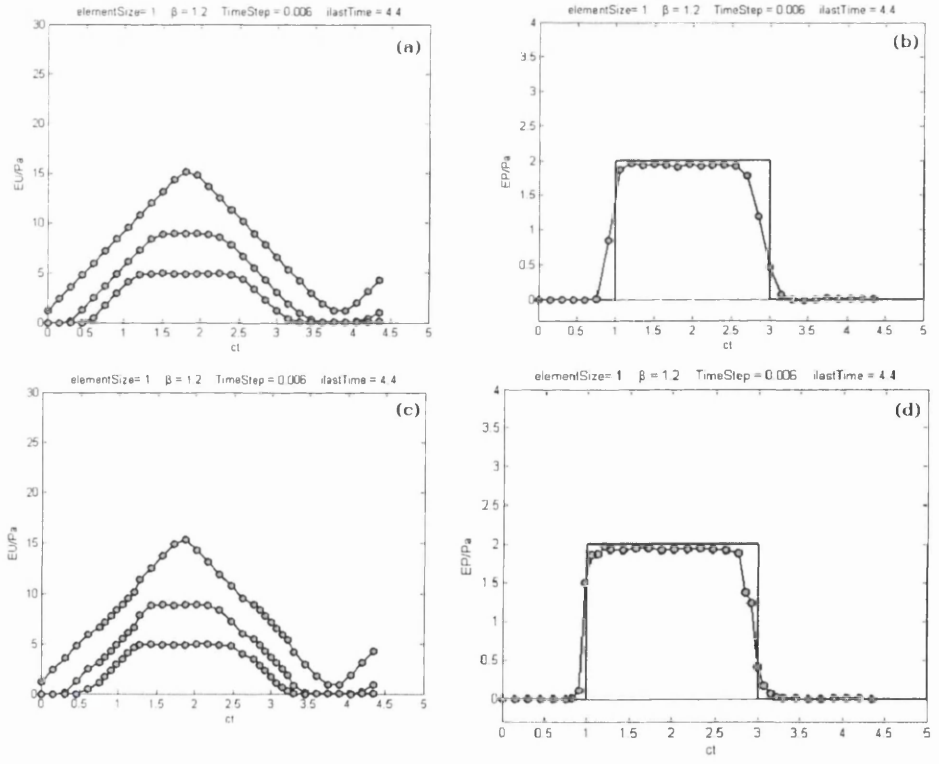


Figure 4.14: Adaptive refinement results:
(a) Displacements at points A, B, C without refinement;
(b) Traction at point D without refinement;
(c) Displacements at points A, B, C with refinement;
(d) Traction at point D with refinement

geomechanics because a semi-infinite domain is involved and the analytical solution is available. Here the surface is discretized into 20 quadratic boundary elements; element size $\triangle h = 7.6m$, time step $\triangle t = 0.03s$, and $\beta = 0.8$. The solution is computed for 50 time steps, and the time history of the vertical displacement at point A is shown in Fig. 4.16. The numerical solution agrees with the analytical solution very well. However, it is interesting to notice that it still remains stable when the time-space ratio β is very small ($\beta = 0.1$), which caused the instability for the wave propagation in the rectangle, as shown in Fig. 4.17(a). It is not difficult to explain why after we examine the discrete BEM formula:

$$H^{nn} \cdot u^n = G^{nn} \cdot q^n + \sum_{m=1}^{n-1} H^{nm} \cdot u^m + G^{nm} \cdot q^m \tag{4.63}$$

Since all elements are on the horizontal plane, $\frac{\partial \mathbf{r}}{\partial \mathbf{n}} = 0$, all elements except the diagonal ones in H^{nn} are zeros; and all elements in H^{nm} are zeros, too. Eq. 4.63 can be rewritten as:

$$\begin{bmatrix} H_{11}^{nn} & \dots & 0 \\ \vdots & H_{22}^{nn} & \vdots \\ & & \\ 0 & \dots & H_{NN}^{nn} \end{bmatrix} \cdot u^n = G^{nn} \cdot q^n + \sum_{m=1}^{n-1} G^{nm} \cdot q^m \tag{4.64}$$

Here q^n and q^m are Heaviside functions and remain at a constant value. G^{nn} is a matrix with the fixed values as well. G^{nm} becomes smaller and smaller at later times. Therefore, there is no source of instability as the time steps decrease.

We also notice that larger β doesn't cause numerical damping (Fig. 4.17(b)). This is simply because that there is no wave front in the boundary elements when the Heaviside function is applied on a half-plane, which are all dissipated as they move towards the infinite boundary. The analytical solution is a slowing changing function Thus, relatively large time steps do not cause any inaccuracies.

Refinement can be applied to this problem, but no obvious advantage can be achieved, as shown in Fig. 4.18. Because it only increases accuracy, which can be achieved by using smaller time steps as well. From this numerical example, we draw the following conclusions:

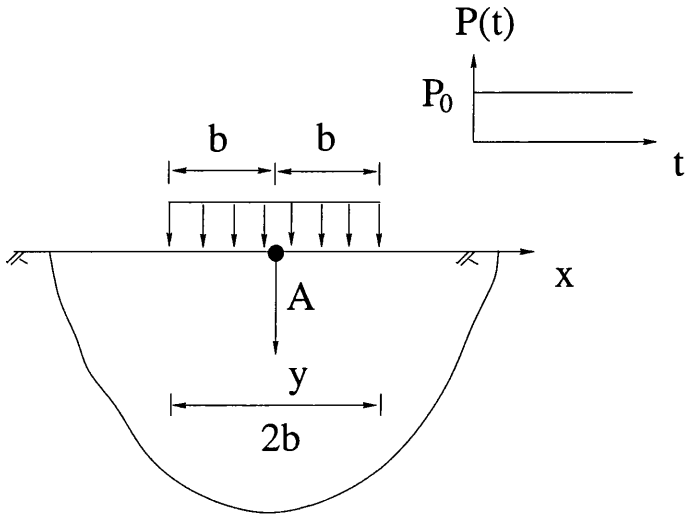


Figure 4.15: Half-plane under transient surface load

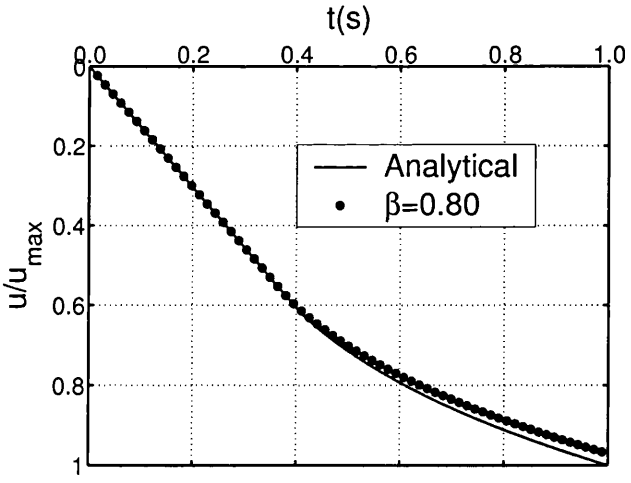
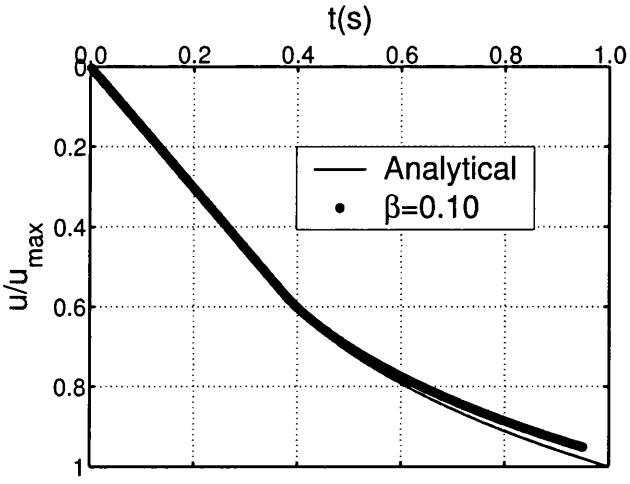
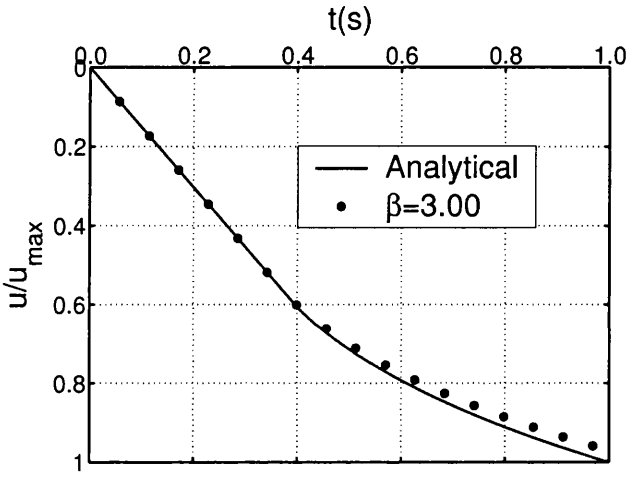


Figure 4.16: Half-plane solution: relative vertical displacement at point A, $\beta = 0.8$



(a)



(b)

Figure 4.17: Half-plane solution: relative vertical displacement at point A, (a) $\beta = 0.1$ (b) $\beta = 3.0$

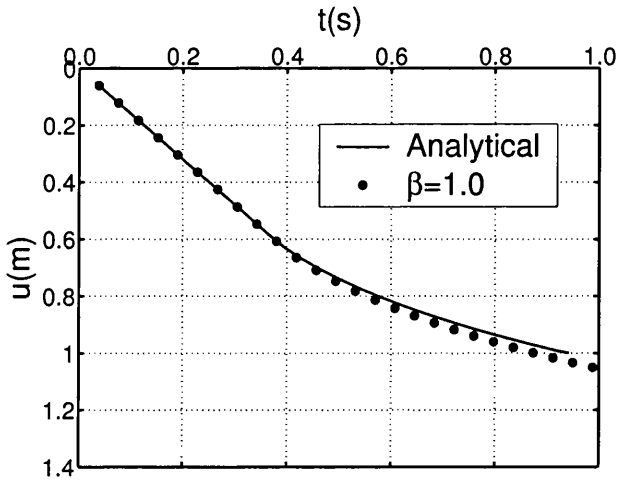


Figure 4.18: Half-plane solution with refinement: relative vertical displacement at point A, $\beta = 1.0$

* In the problem of half-plane subjected to a Heaviside function, because wave energy dissipates in the open domain, the final dynamic response will converge to the static one.

* When the time-space ratio β is very small ($\beta \leq 0.1$ in this case), the solution is still stable; when β is very large ($\beta \geq 3.0$), there is no obvious numerical damping, either. The numerical solution in this example is not sensitive to β values.

* Adaptive schemes lead to more accurate solutions, but without further improvement of its stability.

4.6 Summary

In this chapter, a new boundary integral equation for 2D scalar wave has been derived by the author to accommodate more flexible shape functions in the space-time. The difficulty in BEM solution for impulse wave propagation is to locate the moving high gradient areas which can not be captured by ordinary shape functions in an uniform mesh. The author proposes an adaptive scheme in which moving wave fronts are detected by error estimation in space-time, and a hierarchical adaptive scheme is used to refine the area in order to achieve higher accuracy.

However, 2D Green's function for wave equation is an artificial solution which doesn't represent reality. In the next chapter, we examine the 3D case explicitly.

Chapter 5

Adaptive BEM for scalar wave propagation in 3D

5.1 Introduction

BEM 3D models better describe the true physics of wave phenomena but 3D numerical models are computationally demanding and therefore adaptive schemes are imperative.

This chapter focuses on adaptive numerical implementation of the boundary integral equations for scalar wave propagation in 3D. In Section 5.2, we describe the discretization of BEM formulations of 3D wave equations; spatial descriptions; the evaluation of the time integral, and numerical integration of singular & non-singular kernel functions . In Section 5.3, error estimation and the consequent adaptive schemes for 3D geometry and local time stepping methods are introduced. In Section 5.5, various aspects of BEM programming, such as 3D geometric modeling of arbitrary shapes and their data structure; the memory management of large arrays and matrices; a fast spatial search algorithm; a h- adaptive scheme for 3D mesh refinement and a BEM solver associated with multi-refined meshes, are discussed in detail. Finally, some examples of applications of the adaptive space-time BEM model are discussed in Section 5.6.

5.2 Boundary element method for 3D scalar wave propagation in space-time

5.2.1 Boundary integral formulation for space-time BEM elastodynamics

The governing 3D scalar wave equation is:

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}+\frac{1}{c^2}\mathbf{b}=\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} \tag{5.1}$$

with initial conditions

$$u\left(\mathbf{x},0\right)|_S=u_0\left(\mathbf{x}\right) \qquad \frac{\partial u\left(\mathbf{x},0\right)}{\partial t}|_S=v_0\left(\mathbf{x}\right) \tag{5.2}$$

and boundary conditions

$$u\left(\mathbf{x},t\right)|_{S_1}=\overline{u}\left(\mathbf{x},t\right) \qquad \frac{\partial u\left(\mathbf{x},t\right)}{\partial \mathbf{n}}|_{S_2}=\overline{q}\left(\mathbf{x},t\right) \tag{5.3}$$

where $\mathbf{x}$ denotes spatial coordinates (x,y,z), $\mathbf{n}$ is the unit normal vector on the boundary surface S; on the boundary $S_1 \cup S_2 = S$ and $S_1 \cap S_2 = \emptyset$; $u_0(\mathbf{x})$ and $v_0\left(\mathbf{x}\right)$ are given displacements and velocities on the boundary when time $t=0$; $\overline{u}\left(\mathbf{x},t\right)$ and $\overline{q}\left(\mathbf{x},t\right)$ are prescribed displacements and normal derivatives on the boundary S_1 and S_2 separately at all time. The integral representation for the displacement u at a source point $\mathbf{x}_i$ on the boundary S, at time t with zero body forces and zero initial conditions can be written as (Dominguez [20]):

$$\begin{aligned} c(\mathbf{x}_i)u(\mathbf{x}_i,t) &= \int_0^{t^+}\int_S G(\mathbf{x}_{i,},t-\tau,\mathbf{x})\frac{\partial u}{\partial n}(\mathbf{x},\tau)dS(\mathbf{x})d\tau \\ &- \int_0^{t^+}\int_S \frac{\partial G}{\partial n}(\mathbf{x}_{i,},t-\tau,\mathbf{x})u(\mathbf{x},\tau)dS(\mathbf{x})d\tau \end{aligned} \tag{5.4}$$

where $\boldsymbol{x}$ is a field point on S; τ means the time convolution from 0 to t^+; t^+means to take the upper limit of time t. The Green's function $G(\mathbf{x}_{i,},t-\tau,\mathbf{x})$ is given by

(Dominguez [20]):

$$G(\mathbf{x}_{i,}, t-\tau, \mathbf{x}) = \frac{1}{4\pi r}\delta\left((t-\tau) - \frac{r}{c}\right) \tag{5.5}$$

where $r = |\boldsymbol{x} - \boldsymbol{x}_i|$, c is the wave speed, $\delta(x)$ is the Dirac-delta function.

Substituting the Green's function into Eq. 5.4 , and assuming $q(\mathbf{x}, \tau) = \frac{\partial u(\mathbf{x},\tau)}{\partial n}$, the first time integral of Eq. 5.4 can be written as:

$$\begin{aligned}\int_0^{t^+}\int_S G(\mathbf{x}_{i,}, t-\tau, \mathbf{x}) q(\mathbf{x},\tau) dS(\mathbf{x}) d\tau &= \int_S \frac{1}{4\pi r}\int_0^{t^+} \delta\left((t-\tau) - \frac{r}{c}\right) q(\mathbf{x},\tau) d\tau dS(\mathbf{x}) \\ &= \int_S \frac{1}{4\pi r} q\left(\mathbf{x}, t - \frac{r}{c}\right) dS(\mathbf{x}) \end{aligned} \tag{5.6}$$

In order to compute the second integral in Eq. 5.4, we need to compute the normal derivative of Green's function

$$\begin{aligned}\frac{\partial G}{\partial n}((\mathbf{x}_{i,}, t-\tau, \mathbf{x}) &= \frac{\partial}{\partial n}\left(\frac{1}{4\pi r}\delta\left((t-\tau) - \frac{r}{c}\right)\right) \\ &= \frac{\partial r}{\partial n}\left\{\left(-\frac{1}{4\pi r^2}\delta\left((t-\tau) - \frac{r}{c}\right) + \frac{1}{4\pi r}\frac{\partial}{\partial r}\delta\left((t-\tau) - \frac{r}{c}\right)\right)\right\} \\ &= \frac{\partial r}{\partial n}\left\{\left(-\frac{1}{4\pi r^2}\delta\left((t-\tau) - \frac{r}{c}\right) + \frac{1}{4\pi r c}\frac{\partial}{\partial \tau}\delta\left((t-\frac{r}{c}) - \tau\right)\right)\right\}\end{aligned} \tag{5.7}$$

Thus the second integral in Eq. 5.4 can be written as:

$$\begin{aligned}&\int_0^{t^+}\int_S \frac{\partial G}{\partial n}(\mathbf{x}_{i,}, t-\tau, \mathbf{x}) u(\mathbf{x},\tau) dS(\mathbf{x}) d\tau \\ &= \int_0^{t^+}\int_S \frac{\partial r}{\partial n}\left\{\left(-\frac{1}{4\pi r^2}\delta\left((t-\tau) - \frac{r}{c}\right) + \frac{1}{4\pi r c}\frac{\partial}{\partial \tau}\delta\left((t-\frac{r}{c}) - \tau\right)\right)\right\} u(\mathbf{x},\tau) dS(\mathbf{x}) d\tau \\ &= -\frac{1}{4\pi}\int_S \frac{\partial r}{\partial n}\left\{\left(\frac{1}{r^2}u\left(t - \frac{r}{c}\right) + \frac{1}{rc}\frac{\partial u(\mathbf{x},\tau)}{\partial \tau}\Big|_{\tau = t - \frac{r}{c}}\right)\right\} S(\mathbf{x})\end{aligned} \tag{5.8}$$

Taking account of Eqs. 5.6 and 5.8, the integral representation of the 3D scalar wave equation can be written in the terms of space integrals only (Dominguez [20]):

$$c(\mathbf{x}_i)u(\mathbf{x}_i,t) = \frac{1}{4\pi}\int_S \frac{1}{r}\cdot q(\mathbf{x},t-(r/c))dS(\mathbf{x}) + \frac{1}{4\pi}\int_S \frac{\partial r}{\partial n}\{\frac{1}{r^2}u(\mathbf{x},t-r/c)+\frac{1}{rc}[\frac{\partial u(\mathbf{x},\tau)}{\partial \tau}]_{\tau=t-(r/c)}\}\, dS(\mathbf{x}) \tag{5.9}$$

In Eq. 5.9, the value of u at the point $\mathbf{x}_i$ at time t is determined from the values of $u(\mathbf{x},\tau)$, $q(\mathbf{x},\tau)$ and $\frac{\partial u(\mathbf{x},\tau)}{\partial \tau}$ on the boundary at $\tau = t-(r/c)$. Thus, Eq. 5.9 is called a retarded potential equation because the displacement is determined by what has happened earlier. The space-time integrals of the products of $u(\mathbf{x},\tau)$, $q(\mathbf{x},\tau)$ and Dirac delta function $\delta\left(t-\frac{r}{c}\right)$ become integrals over space only, which surprisingly makes the integral equation of 3D problems much simpler than the 2D counterpart.

Here, the author introduces a new interpolation scheme. Not only is the spatial domain interpolated via shape functions, but also the temporal domain is interpolated using $N^k(\xi,\eta,\tau)$ as a space-time interpolation function:

$$u=\sum N^k(\xi,\eta,\tau)\,u_k; \qquad q=\sum N^k(\xi,\eta,\tau)\,q_k; \qquad t=\sum N^k(\xi,\eta,\tau)\,t_k; \tag{5.10}$$

The distance in space-time is defined as:

$$|ct-r| = \left|c(t_i-\sum N^k(\xi,\eta,\tau)\cdot t_j^k)-r\right| = |c(t_i-\Phi(\xi,\eta,\tau))-r| \tag{5.11}$$

where, $\sum N^k(\xi,\eta,\tau)\cdot t_j^k = \Phi(\xi,\eta,\tau)$.

After disretizing the surface boundary into M surface elements and replacing u and q with their numerical approximation, Eq. 5.9 takes the form of a sum of

integrals of M elements.

$$\begin{aligned} c(\mathbf{x}_i)u(\mathbf{x}_i,t) =& \sum_{m=1}^{M}\left[\int_{S_m}\frac{1}{4\pi}\cdot\frac{1}{r}\cdot\sum N^k(\xi,\eta,\tau)\cdot q_k^j dS(\boldsymbol{x})\right. \\ &+\int_{S_m}\frac{\partial r}{\partial n}\cdot\frac{1}{r^2}\cdot\sum N^k(\xi,\eta,\tau)\cdot u_k^j dS(\boldsymbol{x}) \\ &\left.+\int_{S_m}\frac{\partial r}{\partial n}\cdot\frac{1}{rc}\cdot\left[\frac{\sum N^k(\xi,\eta,\tau)}{\partial\tau}\right]_{\tau=t-(r/c)}\cdot u_k^j dS(\boldsymbol{x})\right] \end{aligned} \tag{5.12}$$

Letting

$$\begin{aligned} G^{ij} &= \textstyle\int_{S_m}\frac{1}{4\pi}\cdot\frac{1}{r}\cdot N^k(\xi,\eta,\tau)\,dS(\boldsymbol{x}) \\ \hat{H}^{ij} &= \textstyle\int_{S_m}\frac{\partial r}{\partial n}\cdot\frac{1}{r^2}\cdot N^k(\xi,\eta,\tau)\,dS(\boldsymbol{x}) \\ &+\textstyle\int_{S_m}\frac{\partial r}{\partial n}\cdot\frac{1}{rc}\cdot\left[\frac{N^k(\xi,\eta,\tau)}{\partial\tau}\right]_{\tau=t-(r/c)}dS(\boldsymbol{x}) \end{aligned} \tag{5.13}$$

and adopting the notation:

$$H^{ij} = \begin{cases} \hat{H}^{ij} & i \neq j \\ \hat{H}^{ij} + c^i & i = j \end{cases} \tag{5.14}$$

the system of equations for all boundary nodes can be expressed in matrix form as

$$\sum_{m=1}^{N}\sum_{j=1}^{M} H^{ij}u^j = \sum_{m=1}^{N}\sum_{j=1}^{M} G^{ij}q^j \tag{5.15}$$

where N is the total number of time steps, and M is the total number of nodes in the space domain. The summation rule applies to j on both sides of the equation.

The causality law requires that boundary values at later times are only influenced by quantities at earlier time, but not *vice versa.* Thus, numerical methods constructed from these space-time boundary integral equations are global in time, i.e., it is necessary to compute the solution for all time steps from the beginning to obtain the current solution. The system is solved step-by-step: once $\mathbf{u}$ and $\mathbf{q}$ are known for the previous time steps, the solution for the *nth* time step is obtained

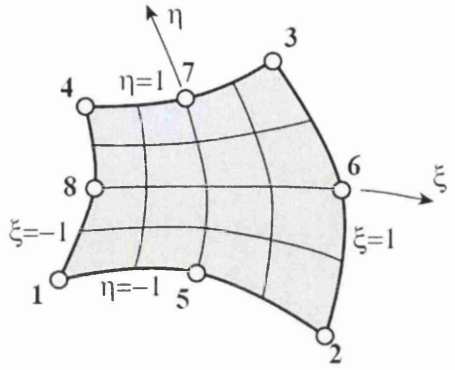


Figure 5.1: Curved quadrilateral element

from:

$$\mathbf{H}^{(nn)ij}\mathbf{u}^{(n)j} = \mathbf{G}^{(nn)ij}\mathbf{q}^{(n)j} + \sum_{m=1}^{n-1} \mathbf{G}^{(nm)ij}\mathbf{q}^{(m)j} - \mathbf{H}^{(nm)ij}\mathbf{u}^{(m)j} \tag{5.16}$$

The boundary consists of the spatial boundary extruded into the space-time, like a cylinder extruded from a 2D circle into 3D space. The increase in dimensionality is offset by special features of the problem. So it retains the same dimensionality as static problems, which are discussed in Section 5.2.3.

5.2.2 Discretization of the 3D scalar wave equation in space

In order to approximate arbitrary shapes and boundary conditions, the geometry and field variables, i.e. coordinates, displacements and tractions, are discretized in the same manner to create the numerical approximation in space.

$$x = \sum N^k(\xi,\eta)\, x_k; \qquad y = \sum N^k(\xi,\eta)\, y_k;$$

$$u = \sum N^k(\xi,\eta)\, u_k; \qquad q = \sum N^k(\xi,\eta)\, p_k; \tag{5.17}$$

For 3D problems, elements commonly used are 6-node triangular elements and 8-node quadrilateral elements.

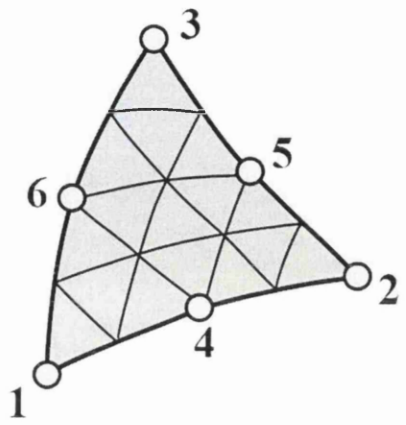


Figure 5.2: curved triangle elements

5.2.2.1 The 8-node curved quadrilateral element

The curved quadrilateral element shown in Fig. 5.1 has shape functions:

$$N_\alpha(\xi,\eta) = \begin{cases} 0.25(1+\xi_0)(1+\eta_0)(\xi_0+\eta_0-1) & if \quad \alpha = 1,3,5,7 \\ 0.50(1+\xi^2)(1-\eta_0) & if \quad \alpha = 2,6 \\ 0.50(1+\xi_0)(1-\eta^2) & if \quad \alpha = 4,8 \end{cases} \tag{5.18}$$

where $\xi_0 = \xi \cdot \xi_\alpha$ and $\eta_0 = \eta \cdot \eta_\alpha$, with ξ and η being the two independent coordinates and $(\xi_\alpha, \eta_\alpha)$ the coordinates of node α as shown in the Fig. 5.1 .

In order to use numerical integral schemes such as Gauss quadrature, Cartesian coordinate system has to be transformed to a local intrinsic coordinate system through Jacobian matrix:

$$J = \frac{\partial(x,y)}{\partial(\xi,\eta)} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial \sum N_\alpha(\xi,\eta)}{\partial \xi} x_\alpha & \frac{\partial \sum N_\alpha(\xi,\eta)}{\partial \xi} y_\alpha \\ \frac{\partial \sum N_\alpha(\xi,\eta)}{\partial \eta} x_\alpha & \frac{\partial \sum N_\alpha(\xi,\eta)}{\partial \eta} y_\alpha \end{bmatrix} \tag{5.19}$$

The Jacobian matrix is the measure of the local element distortion.

5.2.2.2 The 6-node curved triangular element

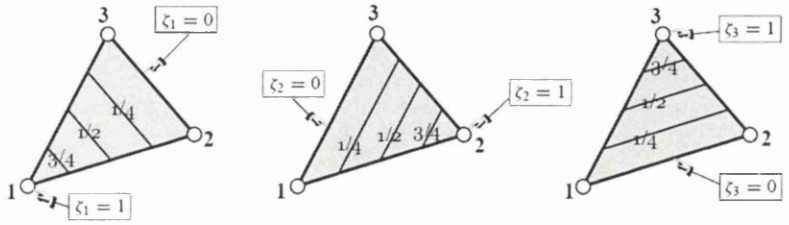


Figure 5.3: Local parametric coordinates of a triangle

The curved triangular element shown in Fig. 5.2 has shape functions:

$$\begin{aligned} N_1 &= \xi_1(2\xi_1 - 1) \\ N_2 &= \xi_2(2\xi_2 - 1) \\ N_3 &= \xi_3(2\xi_3 - 1) \\ N_4 &= 4\xi_1\xi_2 \\ N_5 &= 4\xi_2\xi_3 \\ N_6 &= 4\xi_3\xi_1 \end{aligned} \tag{5.20}$$

where ξ_1, ξ_2, ξ_3 are the local parametric coordinates of the triangle shown in Fig. 5.3.

In order to use numerical integral schemes such as Gauss quadrature, Cartesian coordinate system has to be transformed to a local intrinsic coordinate system through Jacobian matrix:

$$J = \begin{bmatrix} 1 & 1 & 1 \\ \frac{\partial x}{\partial \xi_1} & \frac{\partial x}{\partial \xi_2} & \frac{\partial x}{\partial \xi_3} \\ \frac{\partial y}{\partial \xi_1} & \frac{\partial y}{\partial \xi_2} & \frac{\partial y}{\partial \xi_3} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ \frac{\partial \sum N_\alpha(\xi,\eta)}{\partial \xi_1} x_\alpha & \frac{\partial \sum N_\alpha(\xi,\eta)}{\partial \xi_2} x_\alpha & \frac{\partial \sum N_\alpha(\xi,\eta)}{\partial \xi_3} x_\alpha \\ \frac{\partial \sum N_\alpha(\xi,\eta)}{\partial \xi_1} y_\alpha & \frac{\partial \sum N_\alpha(\xi,\eta)}{\partial \xi_2} y_\alpha & \frac{\partial \sum N_\alpha(\xi,\eta)}{\partial \xi_3} y_\alpha \end{bmatrix} \tag{5.21}$$

5.2.3 Time integral

After the shape functions are defined, the influence matrices $[G]$ and $[H]$ are obtained by integrating over the boundary elements using Eq. (5.13). If the space-time domain $S(\mathbf{x}, t)$ is divided into n parts along the time axis, the integrals for the mth time interval (t_{m-1}, t_m) is the integral over the surface elements that lies within two

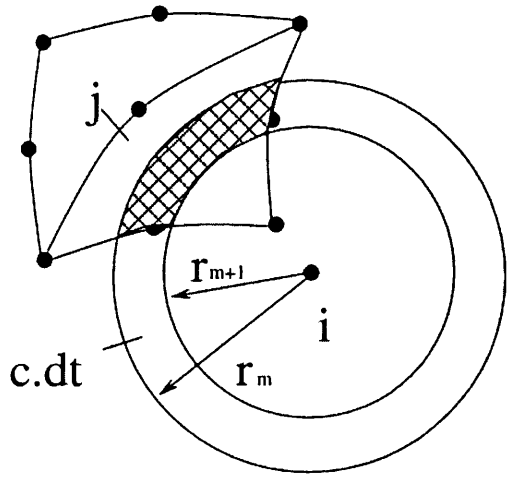


Figure 5.4: Element j receives a signal from the collocation point i during the time interval $t_n - \tau_{m+1} < \frac{r}{c} < t_n - \tau_m$; The coefficients G_{ij}, H_{ij} are integrated over the shaded area only

concentric spherical surfaces of radius $r_m = c(t_n - t_m)$and $r_{m+1} = c(t_n - t_{m+1})$ (Fig. 5.4).

For this reason, the system matrices $[G]$ and $[H]$ are highly sparse. This special structure follows from the convolution of the Dirac delta function in the integral equations. The integration does not extend over the whole boundary of the space-time domain, but only over its intersection with the surface of the backward propagating wave cone. This means that integrals have the same dimensionality as those for static problems, and that current dynamic responses are not affected, in general, by events that extend far into the past. These features are important for 3D scalar wave equations and elastodynamics equations in three dimensions. However, they do not apply to the 2D wave equation. Unlike Dirac delta function in the kernel of 3D scalar wave equations, the 2D kernel influence continue indefinitely.

5.2.4 A fast algorithm for triangle subdivision and global intersection search

This section is the result of a cooperative research endeavour with Dr. Tomasz Koziara, who has expertise in fast spatial searches for contact detection in discrete objects. In the dynamic BEM, we integrate over the intersection of the boundary

surface mesh and spheres with growing radii. The problem is to find those elements which fall into the envelope of the sphere quickly. This is the objective of fast spatial searches.

For an efficient implementation of the computational framework described here, two purely geometrical problems must be addressed. The first one is the *triangle-sphere intersection*, resulting from the integration over the boundary surface contained between the two spheres. The second problem concerns an effective *global search* for the triangle and sphere pairs that intersect.

5.2.4.1 Triangle subdivision

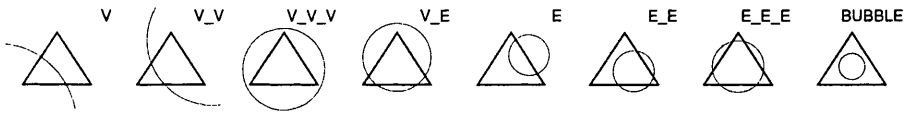


Figure 5.5: Eight out of the nine possible configurations of triangle-sphere intersection (the triangle outside of the sphere case is omitted).

The triangle-sphere intersection problem is resolved by first identifying one of the nine possible intersection configurations (Fig. 5.5), followed by sub-triangulation of the area contained in the sphere. In order to minimize the number of sub-triangles, while keeping a good approximation of the intersection boundary, six-node second order sub-triangles are used. Rather than calculating the triangular area contained between the two spheres, two separate sub-triangulations are obtained by intersecting the triangle with, respectively, the bigger and the smaller sphere. As a result, the terms integrated over the first sub-triangulation are added, while those integrated over the second sub-triangulation are subtracted (Fig. 5.6). This approach is both simpler and more efficient than the explicit derivation of the sub-triangulation contained between the two spheres.

5.2.4.2 Global intersection search

For typical benchmark problems, composed only of a small number of triangles, the issue of efficient identification of the intersecting triangle-sphere pairs can be ignored. This is not true for fine meshes, where the global search for the intersecting pairs

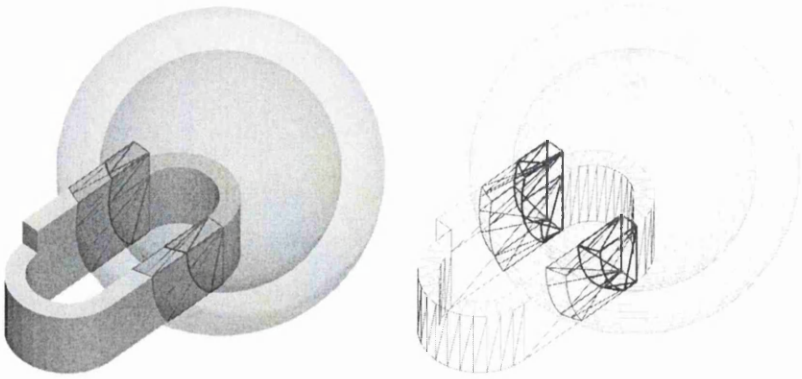


Figure 5.6: Intersection of the triangular mesh of a mechanical component with a pair of spheres. The sub-triangulation boundaries are marked in light grey (terms to be added) and dark grey (terms to be subtracted).

becomes a computational bottleneck. Working with n nodes and m triangles, it is evidently far from optimal to perform a brutal force $O(n \cdot m)$ check, testing all possible pairs of spheres and triangles. The problem at hand falls into the broader category of *spatial search* problems (see [74] for a general introduction), which is common to a range of fields (e.g. contact mechanics, computer graphics, etc.). The specific features here can be summarized as follows

1. The mesh geometry and connectivity does not change during the course of simulation;

2. The search for an intersection takes place within the volume contained between two spheres whose radii increase with time, while their difference (being of an order of the minimal distance between adjacent nodes) remains constant.

The sphere pairs (the *query spheres*) are centered at the mesh nodes. At each time step, the set of query spheres $\mathcal{S}$ intersects a set of surface triangles $\mathcal{T}$. This generates for each node two sub-triangle lists - one for adding and one for subtracting terms during the system matrix assembly. In order to facilitate an efficient identification of intersecting pairs from $\mathcal{S} \times \mathcal{T}$, two techniques were investigated and one of these is advocated here. The first method exploits a *multi-level range and segment tree* structures implemented in a state-of-the-art algorithm HYBRID, presented in [93]. For the reasons to be shown later, this approach proves to be inferior to the one based on a considerably simpler, *sphere-tree* structure [36] (called SPHTREE).

To make use of HYBRID, all objects from the sets $\mathcal{S}$ and $\mathcal{T}$ need to be packed into their *axis aligned bounding boxes*. The algorithm proceeds by processing the two lists of boxes (of query spheres and surface triangles) and recursively building segment and range tree structures along each of the coordinate directions [74, 93]. The gain from the use of recursion is the $O(n+m)$ space utilization[1], while the run-time complexity is $O\left((n+m)\log^3(n+m)+k\right)$, where k is the number of reported box overlaps. This algorithm performs well in many practical cases, although here it cannot show its full strength. Firstly, it is not able to take advantage of the fact that the mesh geometry remains unchanged. A more serious drawback is that when query spheres reach a size comparable to the size of the overall domain, the computational overhead of HYBRID is more significant than that of SPHTREE. Thus, HYBRID performs well only as long as the size of the query spheres remains comparable to the size of the individual surface element.

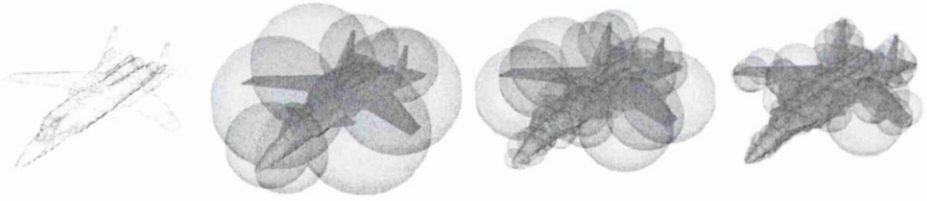


Figure 5.7: Example of a sphere-tree hierarchy built for an airplane surface composed of 10904 triangles.

The SPHTREE approach is better for the current case. At the initial stage of computations, the surface triangulation is wrapped into a sphere-tree hierarchy (Fig. 5.7), which provides the data structure later queried with pairs of spheres. This strategy exploits the fact that the structure remains unchanged during the computation. Here the sphere-tree is built in a simple, top-down manner by the recursive application of a median-plane coordinate bisection along the direction aligned with the longest distribution of surface triangles. At each level of the tree, eight nodes are created, while the last level nodes store no more than sixty four leaf spheres bounding mesh triangles. This tree building strategy is sufficient to show good performance of the sphere-tree based framework, although further research is needed

[1] Conventionally $O(n\log^3 n)$ for a self-intersection test among n boxes [93].

to assess the efficiency of more sophisticated approaches.

The sphere-tree also naturally addresses the issue of finding intersections with a pair of expanding spheres. At each level of the tree, the computations proceed only if the bounding sphere of the current node passes through the volume described by the pair of query spheres. Thus nodes placed inside of the smaller or outside of the bigger of the query spheres are easily omitted. Query traversal is fast, as the necessary numerical tests comprise only a few inexpensive operations (Algorithm 1). The SPHTREE algorithm performs well not only for query spheres with small radii, but also for those of the size comparable to the size of the domain, which is the case for dynamic BEM as here.

Algorithm 1 Surface triangles sphere-tree (T) traversal with a pair of query spheres ($\mathbf{P}$, R_{min}, R_{max}).

```
I = query_spheres_traverse (T,P,R_min,R_max)
1.  d = ||P − T.P||
2.  if d < (R_max + T.R) ∧ d > (R_min − T.R) then
3.    if is_node (T) then
4.      for each Q in T.Children
5.        query_spheres_traverse (Q,P,R_min,R_max)
6.      endfor
7.    else if T.Triangle intersects (P,R_min,R_max) /* it's a leaf */
8.      V = vertex_of (P,R_min,R_max)
9.      T^sub = intersection (T.Triangle,P,R_min)
10.     T^add = intersection (T.Triangle,P,R_max)
11.     I = I ∪ (V,T^sub,T^add)
12.   endif
13. endif
```

5.2.5 Singular integral

If the source node is not within the current element, standard Gauss quadrature is sufficient. Otherwise, singular integrals need careful treatment. For the displacement singularity, the singularity is $O(1/r)$ and can be treated by employing an element subdivision technique to divide the original elements into several triangles. Each triangle subelement can be mapped into square intrinsic element space where the weak singularity is nullified and the integral can be performed using normal Gauss quadrature (Gao and Davies [27]). For the traction singularity of $O(1/r^2)$, an

indirect method is used which employs the rigid body motion condition to calculate the c^i_{lk} coefficient and H^{ii}_{lk}. By looping over each node and element, all terms are calculated and assembled into matrices $[G]$ and $[H]$. Once matrices $[G]$ and $[H]$ have been obtained, the resulting matrix equation is solved by using Gauss elimination or GMRES (Gao and Davies [27]).

5.3 Error indicators and adaptive scheme in 3D

In dynamic BEM, there are three kinds of solution errors, i.e., spatial integration errors, time integration errors and interpolation errors. BEM error analysis by Wenland & Shaw [35] shows that the local space and time errors go to zero if spatial and temporal mesh sizes decrease monotonically. They also show how different local errors interfere with each other and accumulate during time stepping. These analysis help us to choose mathematically correct and practical local error estimators. Without good estimators, computational effort will be wasted due to the poor balancing of space and time errors. Here we confine ourselves to using two simple heuristic, local error indicators. These error indicators, which are cheap and easy to use, will be used to trigger the refinement in high gradient regions.

5.3.1 Gradient-based error indicators

The gradient-based error indicator is based on the assumption that high gradients are strongly localized in space-time, and errors occur in the high-gradient areas. The error indicators are the derivatives of physical quantities within each element, which are normalized by their own pseudo-volumes in space-time.

The error in an element is defined as:

$$e_u = \left[\frac{\int\int \left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial u}{\partial t}\right)^2 \right] dS_e dt}{\int\int dS_e dt} \right]^{\frac{1}{2}}$$

$$e_q = \left[\frac{\int\int \left[\left(\frac{\partial q}{\partial x}\right)^2 + \left(\frac{\partial q}{\partial y}\right)^2 + \left(\frac{\partial q}{\partial t}\right)^2 \right] dS_e dt}{\int\int dS_e dt} \right]^{\frac{1}{2}} \tag{5.22}$$

where u, q are displacements and tractions on the boundary mesh obtained from BEM solutions; x, y are the local curved coordinates on the boundary surface S_e; t is the time.

We substitute the spatial discretization described in Sec. 5.2.2, $u = \sum N^k(\xi,\eta,\tau)\, u_k$; $q = \sum N^k(\xi,\eta,\tau)\, q_k$, and we apply some transformations:

$$\begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial t} \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} & \frac{\partial \tau}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} & \frac{\partial \tau}{\partial y} \\ \frac{\partial \xi}{\partial t} & \frac{\partial \eta}{\partial t} & \frac{\partial \tau}{\partial t} \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial \xi} \\ \frac{\partial u}{\partial \eta} \\ \frac{\partial u}{\partial \tau} \end{bmatrix} \tag{5.23}$$

$$\begin{aligned} \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial u}{\partial t}\right)^2 &= \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial t} \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial t} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial u}{\partial \xi} & \frac{\partial u}{\partial \eta} & \frac{\partial u}{\partial \tau} \end{bmatrix} \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} & \frac{\partial \tau}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} & \frac{\partial \tau}{\partial y} \\ \frac{\partial \xi}{\partial t} & \frac{\partial \eta}{\partial t} & \frac{\partial \tau}{\partial t} \end{bmatrix}^T \cdot \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} & \frac{\partial \tau}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} & \frac{\partial \tau}{\partial y} \\ \frac{\partial \xi}{\partial t} & \frac{\partial \eta}{\partial t} & \frac{\partial \tau}{\partial t} \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial \xi} \\ \frac{\partial u}{\partial \eta} \\ \frac{\partial u}{\partial \tau} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial u}{\partial \xi} & \frac{\partial u}{\partial \eta} & \frac{\partial u}{\partial \tau} \end{bmatrix} \left(J^{-1}\right)^T J^{-1} \begin{bmatrix} \frac{\partial u}{\partial \xi} & \frac{\partial u}{\partial \eta} & \frac{\partial u}{\partial \tau} \end{bmatrix}^T \end{aligned} \tag{5.24}$$

where we assume

$$J^{-1} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} & \frac{\partial \tau}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} & \frac{\partial \tau}{\partial y} \\ \frac{\partial \xi}{\partial t} & \frac{\partial \eta}{\partial t} & \frac{\partial \tau}{\partial t} \end{bmatrix} \tag{5.25}$$

since

$$\begin{bmatrix} \frac{\partial u}{\partial \xi} \\ \frac{\partial u}{\partial \eta} \\ \frac{\partial u}{\partial \tau} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial t}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial t}{\partial \eta} \\ \frac{\partial x}{\partial \tau} & \frac{\partial y}{\partial \tau} & \frac{\partial t}{\partial \tau} \end{bmatrix} \begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial t} \end{bmatrix} \tag{5.26}$$

We define

$$J = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial t}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial t}{\partial \eta} \\ \frac{\partial x}{\partial \tau} & \frac{\partial y}{\partial \tau} & \frac{\partial t}{\partial \tau} \end{bmatrix} \tag{5.27}$$

also

$$dxdydt = |J|\, d\xi d\eta d\tau \tag{5.28}$$

Hence:

$$e_u = \frac{\int \begin{bmatrix} \frac{\partial u}{\partial \xi} & \frac{\partial u}{\partial \eta} & \frac{\partial u}{\partial \tau} \end{bmatrix} (J^{-1})^T J^{-1} \begin{bmatrix} \frac{\partial u}{\partial \xi} & \frac{\partial u}{\partial \eta} & \frac{\partial u}{\partial \tau} \end{bmatrix}^T |J|\, d\xi d\eta d\tau}{\int |J|\, d\xi d\eta d\tau}$$

$$e_q = \frac{\int \begin{bmatrix} \frac{\partial q}{\partial \xi} & \frac{\partial q}{\partial \eta} & \frac{\partial q}{\partial \tau} \end{bmatrix} (J^{-1})^T J^{-1} \begin{bmatrix} \frac{\partial q}{\partial \xi} & \frac{\partial q}{\partial \eta} & \frac{\partial q}{\partial \tau} \end{bmatrix}^T |J|\, d\xi d\eta d\tau}{\int |J|\, d\xi d\eta d\tau} \tag{5.29}$$

where

$$\begin{bmatrix} \frac{\partial u}{\partial \xi} & \frac{\partial u}{\partial \eta} & \frac{\partial u}{\partial \tau} \end{bmatrix} = \begin{bmatrix} \frac{\partial \sum N^k(\xi,\eta,\tau) u_k}{\partial \xi} & \frac{\partial \sum N^k(\xi,\eta,\tau) u_k}{\partial \eta} & \frac{\partial \sum N^k(\xi,\eta,\tau) u_k}{\partial \tau} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial q}{\partial \xi} & \frac{\partial q}{\partial \eta} & \frac{\partial q}{\partial \tau} \end{bmatrix} = \begin{bmatrix} \frac{\partial \sum N^k(\xi,\eta,\tau) q_k}{\partial \xi} & \frac{\partial \sum N^k(\xi,\eta,\tau) q_k}{\partial \eta} & \frac{\partial \sum N^k(\xi,\eta,\tau) q_k}{\partial \tau} \end{bmatrix}$$

$N^k(\xi, \eta, \tau)$ are the shape functions used in the space-time domain; u_k and q_k are errors of nodal displacements and tractions on the boundary and $|J(\xi, \eta)|$ is the determinant of the Jacobian obtained from Eq. 5.27. Assuming that curved triangular elements are used in space and linear elements are used in time, the exact formula for error estimation can be derived from these formulas. The displacement is:

$$u = \frac{\tau - \tau_{m-1}}{\tau_m - \tau_{m-1}} \sum_{k=1}^{6} N^k(\xi, \eta) u_m^k + \frac{\tau_m - \tau}{\tau_m - \tau_{m-1}} \sum_{k=1}^{6} N^k(\xi, \eta) u_{m-1}^k \tag{5.30}$$

which yields:

$$\frac{\partial u}{\partial \xi} = \frac{\tau - \tau_{m-1}}{\tau_m - \tau_{m-1}} \sum_{k=1}^{6} \frac{\partial N^k(\xi, \eta)}{\partial \xi} u_m^k + \frac{\tau_m - \tau}{\tau_m - \tau_{m-1}} \sum_{k=1}^{6} \frac{\partial N^k(\xi, \eta)}{\partial \xi} u_{m-1}^k$$

$$\frac{\partial u}{\partial \eta} = \frac{\tau - \tau_{m-1}}{\tau_m - \tau_{m-1}} \sum_{k=1}^{6} \frac{\partial N^k(\xi,\eta)}{\partial \eta} u_m^k + \frac{\tau_m - \tau}{\tau_m - \tau_{m-1}} \sum_{k=1}^{6} \frac{\partial N^k(\xi,\eta)}{\partial \eta} u_{m-1}^k$$

$$\frac{\partial u}{\partial \tau} = \frac{\tau}{\tau_m - \tau_{m-1}} \sum_{k=1}^{6} N^k(\xi,\eta) u_m^k - \frac{\tau}{\tau_m - \tau_{m-1}} \sum_{k=1}^{6} N^k(\xi,\eta) u_{m-1}^k \tag{5.31}$$

where the derivatives of the 2nd order shape functions of the triangle element with respect to the local coordinates are:

$$\frac{\partial N^k(\xi,\eta)}{\partial \xi} = \left[\begin{array}{ccccccc} -1+4\xi, & 0, & -3+4\xi+4\eta, & 4\eta, & -4\xi, & -4(-1+\eta+2\xi) \end{array} \right]$$

$$\frac{\partial N^k(\xi,\eta)}{\partial \eta} = \left[\begin{array}{ccccccc} 0, & -1+4\eta, & -3+4\xi+4\eta, & 4\xi, & -4(-1+2\eta+\xi), & -4\xi \end{array} \right] \tag{5.32}$$

Since time t is independent of the local curved space coordinates (x, y), the determinant of the Jacobian is:

$$|J| = \left| \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & 0 \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & 0 \\ 0 & 0 & 1 \end{bmatrix} \right| = \left| \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \right| \tag{5.33}$$

Thus the inverse of the Jacobian is

$$J^{-1} = \frac{1}{|J|} \begin{bmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial y}{\partial \xi} \\ -\frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \xi} \end{bmatrix} \tag{5.34}$$

where

$$|J| = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \eta} \tag{5.35}$$

5.3.2 Re-solution error estimation

Besides deriving errors from high gradients of boundary values, error estimates can also be obtained from the difference between coarse and refined meshes. The definition of the error indicator for displacements u is the difference between the gradients $\triangle\frac{\partial u}{\partial xi} = \frac{\partial u}{\partial xi}|_{new} - \frac{\partial u}{\partial xi}|_{old}$, because we assume that errors only exist when two displacement field's gradients are different. The error indicator for traction q is $\triangle q = q_{new} - q_{old}$. q_{new} are the values from the refined elements while q_{old} are the values from the coarse mesh. The formulas for their error estimates are:

$$e_u = \left(\frac{\int\int \left[\left(\triangle\frac{\partial u}{\partial x}\right)^2 + \left(\triangle\frac{\partial u}{\partial y}\right)^2 + \left(\triangle\frac{\partial u}{\partial t}\right)^2 \right] dS_e dt}{\int\int dS_e dt} \right)^{\frac{1}{2}}$$

$$e_q = \left(\frac{\int\int (\triangle q)^2 \, dS_e dt}{\int\int dS_e dt} \right)^{\frac{1}{2}} \tag{5.36}$$

In discrete form:

$$e_u = \left(\frac{\int \left[\begin{array}{ccc} \frac{\partial\triangle u}{\partial\xi} & \frac{\partial\triangle u}{\partial\eta} & \frac{\partial\triangle u}{\partial\tau} \end{array} \right] (J^{-1})^T J^{-1} \left[\begin{array}{ccc} \frac{\partial\triangle u}{\partial\xi} & \frac{\partial\triangle u}{\partial\eta} & \frac{\partial\triangle u}{\partial\tau} \end{array} \right]^T |J(\xi,\eta)| \, d\xi d\eta d\tau}{\int |J(\xi,\eta)| \, d\xi d\eta d\tau} \right)^{\frac{1}{2}}$$

$$e_q = \left(\frac{\int (\triangle q)^2 \, |J(\xi,\eta)| \, d\xi d\eta d\tau}{\int |J(\xi,\eta)| \, d\xi d\eta d\tau} \right)^{\frac{1}{2}} \tag{5.37}$$

where all the terms are the same as the last section except that
the displacement vector $\left[\begin{array}{cccccc} u_m^1 & u_m^2 & u_m^3 & u_m^4 & u_m^5 & u_m^6 \end{array} \right]^T$
and the traction vector $\left[\begin{array}{cccccc} q_m^1 & q_m^2 & q_m^3 & q_m^4 & q_m^5 & q_m^6 \end{array} \right]^T$
are changed to:
$\left[\begin{array}{cccccc} \triangle u_m^1 & \triangle u_m^2 & \triangle u_m^3 & \triangle u_m^4 & \triangle u_m^5 & \triangle u_m^6 \end{array} \right]^T$
and $\left[\begin{array}{cccccc} \triangle q_m^1 & \triangle q_m^2 & \triangle q_m^3 & \triangle q_m^4 & \triangle q_m^5 & \triangle q_m^6 \end{array} \right]^T$.

The advantage of this strategy is that we combine the error estimate and the adaptivity together. We continue the refinement of elements where errors are large

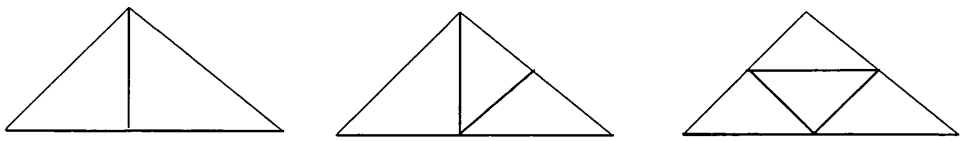


Figure 5.8: Refine a triangle into (from left to right) into two, three and four subtriangles.

while halting the refinement in elements where errors are small.

5.3.3 The triangulation refinement based longest edge propagation path (LEPP)

Once the elements with the largest errors are found, they are refined to reduce discretization errors. Triangular elements in the surface can be refined into two, three or four subtriangles as shown in Fig. 5.8. All these divisions are based on the strategy of dividing the longest edge. In Fig. 5.8, the bottom edge of the original triangle is the longest one and it is halved in any refinement.

Here the author uses the triangulation refinement scheme based on the Longest-Edge Propagation Path (LEPP) of a triangular mesh (M. Rivara and N. Hitschfeld [69]). For any triangular element t_0 of any conforming triangular mesh, the Longest-Edge Propagation Path of t_0 is the ordered list of all the triangle elements $t_0, t_1, t_2, ...t_n$, such that t_i is the neighbour triangle of t_{i-1} by the longest edge of t_{i-1}, for $i = 1, 2, ..., n$. The ordered list of all the triangle elements $t_0, t_1, t_2, ...t_n$ is denoted as $LEPP(t_0)$. Two adjacent triangles (t_i, t_{i-1}) is called a pair of terminal triangles if they share the longest edge in the triangle list. In addition, t_i itself alone will be a terminal boundary triangle if its longest-edge lies on the geometry boundary. In Fig. 5.9, the Longest-Edge Propagation Path of t_0 is the ordered list of triangles (t_0, t_1, t_2, t_3); the pair (t_2, t_3) is a pair of terminal triangles.

Fig. 5.9 also illustrates the process of the refinement of the triangle t_0 based on LEPP. First we find the Longest-Edge Propagation Path of t_0 is the ordered list of triangles (t_0, t_1, t_2, t_3). The terminal triangles (t_2, t_3) are refined by the bisection of the longest common edge. Then the new triangle list of the Longest-Edge Propagation Path of t_0 is set up and the last pair is bisected again. The Longest-Edge

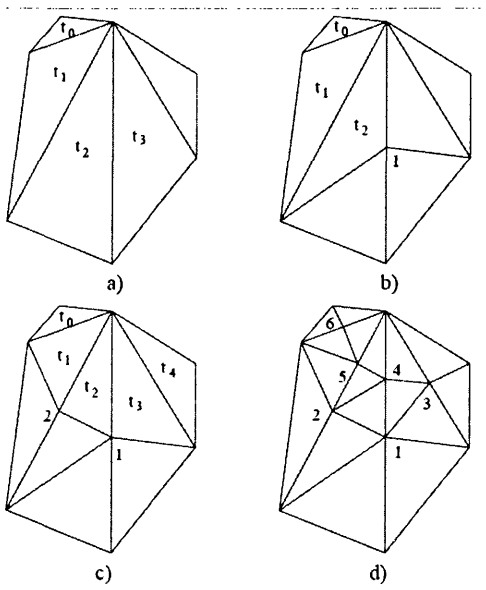


Figure 5.9: LEPP refinement of triangle t_0 (a) Initial triangulation (b) First step of the process (c) Second of the process (d) Final triangulation

Propagation Path of the triangle t_0 is repeatedly built up over the current mesh in order to find the last two terminal triangles to refine, until the initial triangle t_0 is refined.

It can be mathematically shown [69] that the LEPP refinement scheme guarantees the construction of good-quality refined meshes with linear time complexity, provided that an initial good quality triangular mesh is used. In order to implement this algorithm, a suitable data structure that explicitly manages the neighbour-triangle relation is used, which will be described in more detail in Section 5.5.2. The algorithm of LEPP refinement is described in Algorithm 2.

Algorithm 2 LEPP-refinement procedures

```
LEPP_Refine (t,T)
1. while t remains without being bisected do
2.  Find the LEPP(t)
3.  if t_n, t_{n-1}, the last pair of the LEPP(t) are terminal triangles
4.  then
5.    bisect t_n, t_{n-1}
6.  else
7.    bisect t_n  only
8.  endif
9. end while
```

5.3.4 Adaptive algorithm

The automatic mesh refinement strategies for 3D scalar wave problems can be summarized as:

$$SOLVE \Rightarrow ESTIMATE \Rightarrow MARK \Rightarrow REFINE$$

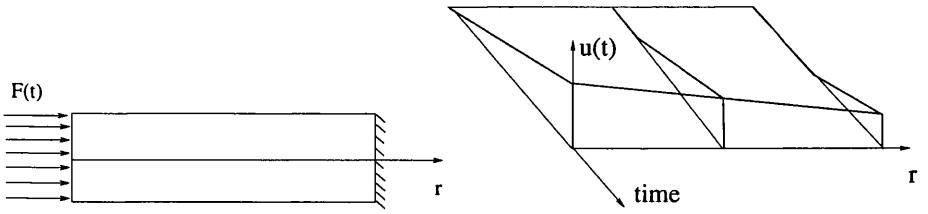
Given the initial triangulation of surface boundaries, one has to compute the BEM solution in the first step $SOLVE$. The solution error is estimated by postprocessing the solution results from the last step: this is the second step, $ESTIMATE$. On the basis of the refinement indicators derived from the error estimate, the step $MARK$ identifies the elements in the current mesh in need of refinement. The new mesh is generated in the last step $REFINE$ and sent to $SOLVE$ step in the next loop. The iteration continues until the BEM solution satisfies the prescribed error tolerance. The general adaptive process is similar to the one in the Chapter. 4.

5.4 Local time stepping

Usually, BEM employs a finite difference methodology in time and boundary element discretization in space. This means that the same time step is applied for all elements. However, for impulse wave problems, regions of high stress or high strain evolve over time, and rapidly moving regions of high stress gradients exhibit a high degree of localization at any instant in time. Using smaller time steps for all regions is computationally wasteful. Thus, it is more efficient to build a BEM model with local time stepping, combined with an adaptive scheme which indicates where it should apply.

5.4.1 Objectives of local time stepping

Here a simple example of a rod subjected to Heaviside load is used to demonstrate local time stepping. The configuration of the problem and the moving wavefront in space-time are shown in Fig. 5.10. Since it takes different times for the wave front to reach different points along the rod, the displacements at different points will be



(a) (b)

Figure 5.10: (a) Heaviside load on a rod; (b) moving wavefront in space-time

quiescent at various times (shown in Fig. 5.10) .

In the integral representation formula for 3D wave problems (Eq. 5.38), the numerical approximation of $\frac{\partial u(\mathbf{x},\tau)}{\partial \tau}$ causes significant problems if the time step is too large.

$$
\begin{aligned}
c(\mathbf{x}_i)u(\mathbf{x}_i,t) &= \frac{1}{4\pi}\int_S \frac{1}{r}\cdot q(\mathbf{x},t-(r/c))dS(\mathbf{x}) \\
&+ \frac{1}{4\pi}\int_S \frac{\partial r}{\partial n}\{\frac{1}{r^2}u(\mathbf{x},t-r/c)+\frac{1}{rc}[\frac{\partial u(\mathbf{x},\tau)}{\partial \tau}]_{\tau=t-(r/c)}\}\, dS(\mathbf{x})
\end{aligned}
\tag{5.38}
$$

For example, if $u(\mathbf{x},\tau)$ is approximated linearly in time and quadratically in space (any higher order approximation will have similar problems), the numerical approximation of $u(\mathbf{x},\tau)$ and its time derivative are:

$$
u(\mathbf{x},\tau) = \frac{\tau-\tau_{m-1}}{\tau_m-\tau_{m-1}}u(\mathbf{x})
$$

$$
\frac{\partial u(\mathbf{x},\tau)}{\partial \tau} = \frac{1}{\tau_m-\tau_{m-1}}u(\mathbf{x}) \tag{5.39}
$$

Substituting these expressions into the representation formula, the third term in the integral of Eq. 5.38 is

$$
I_3 = \frac{1}{4\pi}\int_S \frac{\partial r}{\partial n}\{\frac{1}{rc}\frac{1}{\tau_m-\tau_{m-1}}u(\mathbf{x})\}\, dS(\mathbf{x}) \tag{5.40}
$$

In reality, the term $\frac{\partial u(\mathbf{x},\tau)}{\partial \tau} = c$. However, in the linear time approximation scheme, the discontinuities of $u(\mathbf{x},\tau)$ is not considered in the approximation. Thus,

$\frac{\partial u(\mathbf{x},\tau)}{\partial \tau} = \frac{1}{\tau_m - \tau_{m-1}} u(\mathbf{x})$ instead. This term can produce large errors when the time step is large. This conclusion holds for any impulse loading according to the same reasoning. Of course, this problem can be alleviated by applying smaller time steps to better resolve the rapid change of displacement in time. Unfortunately, since the wavefront is moving, the high gradient of $\frac{\partial u(\mathbf{x},\tau)}{\partial \tau}$ occurs in different locations at different times. If global time stepping is used, although the refined time step is needed in only a small area, the time step will be refined for all elements, and this is wasteful from the computational point of view. After local time stepping is introduced, only the time step of high gradient areas will be refined while others will retain large time steps.

5.4.2 Local time stepping scheme and implementation

The fully explicit algorithm of local time stepping for BEM proposed in this section was developed by the author, inspired by a paper in the FEM context (S. Piperno [67]). In that paper, a symplectic local time-stepping for the arbitrarily unstructured FEM mesh is used to solve the wave diffraction problem. The set of elements are partitioned into N classes according to their different sizes, the global time step of algorithm is $\triangle t$; for $1 \leq k \leq N$, elements of the class k will be time-advanced using the *leap frog* method with the local time step $\triangle t/2^{N-k}$; the largest elements are in class N and the smallest in class 1. The time stepping rotates among the different classes, in a zigzag fashion, until all of them reach the next time step t^{n+1} from t^n.

However, there are some complications in local time-stepping for adaptive BEM which are different from FEM:

1. The elements to be refined are unknown before the computation. They are different at each time step, and have to be decided by error estimation after postprocessing the BEM solution. The partition of elements can not be done once and for all; the classification of elements should be dynamically updated at each time step.

2. FEM in dynamics usually involves physical values at the last time step, or past several steps at most for the multi-time-step case. As a global operator,

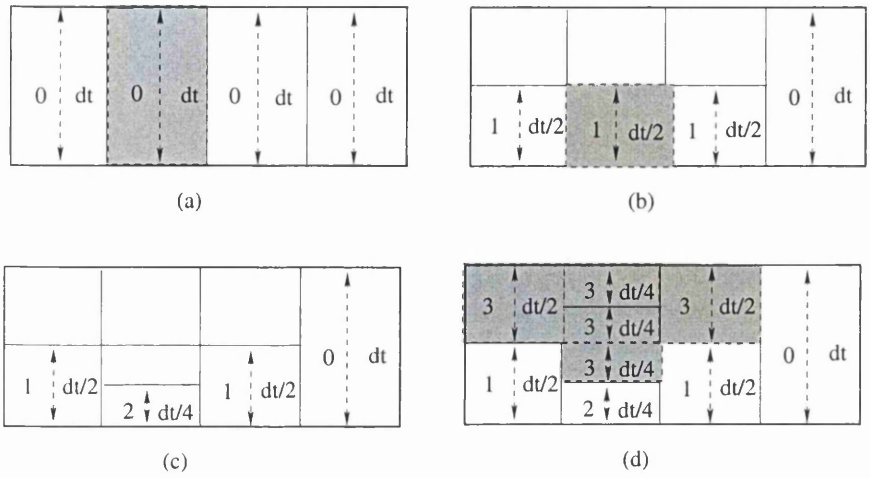


Figure 5.11: Local time stepping: sub-steps are performed from step 0 to step 3

dynamic BEM involves physical values at all past times. Thus local time stepping should be carefully designed to allow for this.

The new local time stepping proposed by the author was designed with consideration for these differences from the FEM. Let us firstly assume that:

1. The set of elements are partitioned into N classes. This partition will be dynamically updated at each time step based on the error estimation of BEM solutions and on geometrical or physical criteria.

2. The global time step of the algorithm is $\triangle t$; Elements of the class k $(0 \leq k \leq N)$ will be time-advanced using the integral representation formula of BEM with the local time step $\triangle t/2^k$; the largest elements are in class 0 and the smallest in class N.

The algorithm is schematically illustrated in Fig. 5.11 and is described in more detail as follows:

Let us denote the algorithm $R^N(\tau)$ for advancing classes N in time over the time interval $\tau > 0$ (If we start from a uniform mesh, all elements belong to class 0. The elements after the first refinement belong to class 1; those after second refinement belong to class 2, and so on). We define $R^N(\tau)$ in a recursive way. Since the uniform mesh advancing in global time step $\triangle t$ using ordinary BEM is called $R^0(\tau)$, for any $N \geq 0$, the $R^{N+1}(\tau)$ is defined as follows:

1. Start with boundary values $u(\mathbf{x})$ and $q(\mathbf{x})$ which are known at time $t^n = n\triangle t$;

2. Advance all elements in class $k \leq N$ with $R^N(\triangle t)$ to obtain all unknown boundary values $u(\mathbf{x})$ and $q(\mathbf{x})$ at time $t^{n+1} = (n+1)\triangle t$; Use gradient-based error estimation to mark those elements with high gradient values, which are labeled as class $N+1$.

3. Refine the elements in class $N+1$, Advance all elements in class $N+1$ with $R^N(\triangle t/2)$, see Fig. 5.11 (b). If required, use values at time t^{n+1} for elements in class N.

4. Since the elements in class $N+1$ are solved twice so far, use the two solution error estimates to mark those elements with large errors, which are labeled as class $N+2$.

5. Refine the elements in class $N+2$, Advance all elements in class $N+2$ with $R^N(\triangle t/4)$, see Fig. 5.11 (c). If required, use values at time t^{n+1} for elements in class $k \leq N+1$.

6. Continue this process of two solution error estimates, refinement in both space and time to be class k and advance the element in class k with $R^N(\triangle t/2^k)$, until the error tolerance is satisfied.

7. After the high-gradients of the boundary values $u(\mathbf{x})$ and $q(\mathbf{x})$ are located and refinements are made, for all element in class $k > 0$, finish the remaining time with $R^N(\triangle t/2^k)$ in one step until t^{n+1} is reached. For an example shown in Fig. 5.11 (d), we can write down the boundary integral equations in space-time for elements in class 2 to finish another three time steps ($\triangle t/4$), then elements in class 1 in another $\triangle t/2$ time step to reach t^{n+1}(which are all marked in grey).

8. All unknowns at time t^{n+1} have been computed with local time stepping as well as satisfying the prescribed tolerance for the solution error.

5.5 Numerical Implementation

In this section, the numerical implementation using the C language is explained in outline. The computer program discussed here may be used to solve 3D scalar wave propagation problems for arbitrary geometry and boundary conditions.

5.5.1 Program structure

The adaptive boundary element program can be separated into five basic parts:

1. Geometry and boundary condition input module, mesh preprocessor
2. Compute the G_{ij}^e and H_{ij}^e— influence matrices of each element, assemble into general matrices $[G]$ and $[H]$.
3. Linear system solver
4. Error estimation and marking those elements with large errors
5. Boundary element refinement

Fig. 5.12 shows a schematic flow-chart for the adaptive boundary element program. Each of the modules is very complex in practice. In the subsequent sections we will discuss the programming aspects for each of these modules.

The implementation process is divided into 7 subsections: Section 5.5.2 describes the basic data structure for surface vertex, triangles and surface meshes in 3D, which are used to describe objects with arbitrary shape. Since 3D BEM models may involve hundreds or thousands of nodes and elements, Section 5.5.3 describe the memory management codes for large arrays and large list structures. In order to do a fast search for intersections between spheres and boundary surfaces, surface meshes are enveloped in a special data structure called sphere trees. How to construct such a sphere tree and how it functions is described in Section 5.5.4. Section 5.5.5 discusses various other issues, such as computing the influence matrices of each element, G_{ij}^e and H_{ij}^e; assembling them into the general matrices $[G]$ and $[H]$ and linear system solvers. In Section 5.5.7, the mesh refinement strategy and its implementation are described. Various error estimation methods are described in Section 5.5.6 and

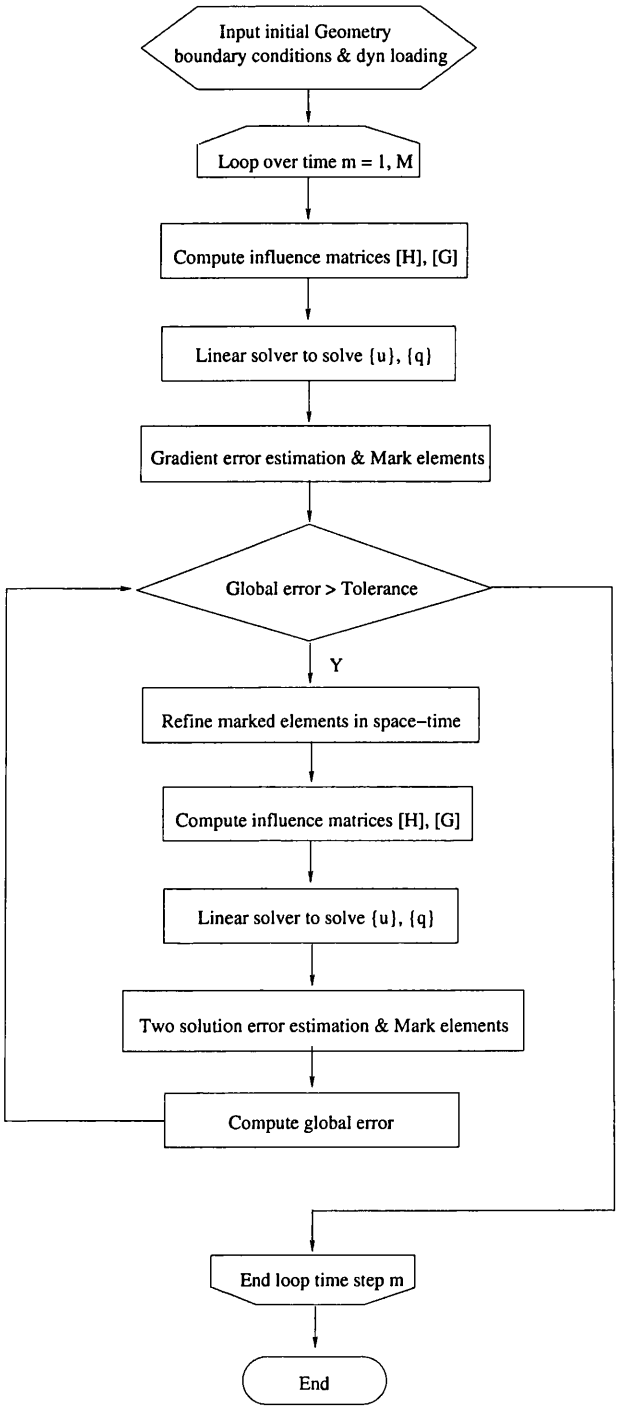
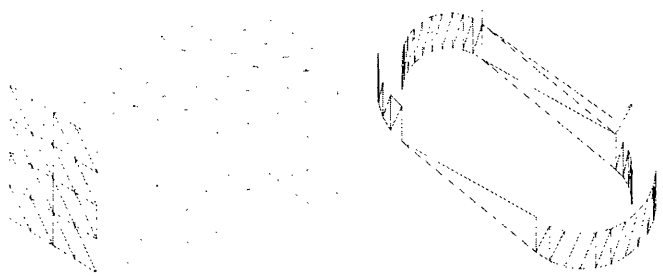


Figure 5.12: Simplified scheme of adaptive boundary element program



(a) (b)

Figure 5.13: Some objects modeled by triangular elements. (a) a cube; (b) a mechanical part

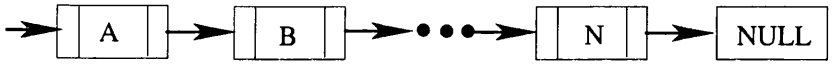


Figure 5.14: Double direction list structure

finally Section 5.5.8 describes the changes needed in the BEM solver after introducing the new adaptive meshes.

5.5.2 Geometry modeling - surface mesh and data structures

The geometry of surface boundaries are modeled by triangles in 3D. The triangular element is chosen for its ability to approximate any geometric shape in 3D while quadrilateral elements fail in certain cases. Some objects modeled by triangular elements are shown in Fig. 5.13.

The two main functions of a mesh are: i) describing the locations of nodes. ii) the relationship between nodes and elements, i.e., which nodes belong to which element. Data structures of the surface mesh mainly consist of two list structures: the double-direction list (as shown in Fig. 5.14) of vertexes to describe the coordinates of nodes, the double-direction list of triangles to describe node-element relations, and the structure of a mesh which includes the two lists above.

The double-direction list of vertexes includes its 3D coordinates, boundary values, its own number in the vertex list, the radius for two intersection spheres, two list pointers for sub-triangles to be added or subtracted, and list pointers to the previous one and to the next one, as shown in Algorithm 3.

The double-direction list of triangles consists of a 6-element array: it has a

Algorithm 3 Data structure of the surface vertex

```
struct surface_vertex
{
double coord [3];              // vertex coordinates
Surface_Boundary_Value *sbv;   // Vertex boundary values
unsigned int number;           // vertex number
double rmin, rmax;             // Radii of intersection spheres
Surface_Sub_Triangle *add;     // list of sub-triangles to be added
Surface_Sub_Triangle *sub;     // list of sub-triangles to be subtracted
Surface_Vertex *prev, *next;   // list structure
};
```

Algorithm 4 Data structure of the surface triangle

```
struct surface_triangle
{
Surface_Vertex *ver [6];       // vertexes
Surface_Boundary_Value *sbv    // Vertex boundary values ;
Surface_Triangle *nei [3];     // neighbors
unsigned int number;           // triangle number
Surface_Triangle *prev, *next;// list structure
};
```

pointer of surface vertexes which indicate which nodes belong to which element; triangle boundary values, three pointers to neighbouring triangles (which is useful for LEPP-based mesh refinement and postprocessing of stresses etc.); triangle number and list pointers to the previous and to the next triangle, as shown in Algorithm 4.

The data structure of the surface mesh consists of the order of the triangle (either 3-node linear elements or 6-node quadratic elements); several memory-pools (whose function will be explained in the next section); the total number of vertexes; a pointer to the list of vertexes; the total number of triangles; a pointer to the list of triangles, and a pointer to the sphere tree (which will be explained in Section 5.5.4).

5.5.3 Memory management for large arrays

Because BEM 3D models usually involve thousands of vertexes and surface triangles, it is essential to deal efficiently with large memories in which the same data structure repeats many times. A *memory pool* allows dynamic memory allocation such as *malloc* in C or the operator *new* in C++. The implementations suffer from memory

Algorithm 5 Data structure of the surface mesh

```
struct surface_mesh
{
short order;              // order of triangles
struct mempool vpool;     // vertexes pool
struct mempool tpool;     // triangles pool
struct mempool ipool;     // intersection triangles pool
struct mempool spool;     // sphere tree pool
struct mempool bpool;     // boundary values pool
unsigned int nver;        // vertexes count
Surface_Vertex *ver;      // list of vertexes
unsigned int ntri;        // triangles count
Surface_Triangle *tri;    // list of triangles
Sphere_Tree *tree;        // sphere tree for triangles
};
```

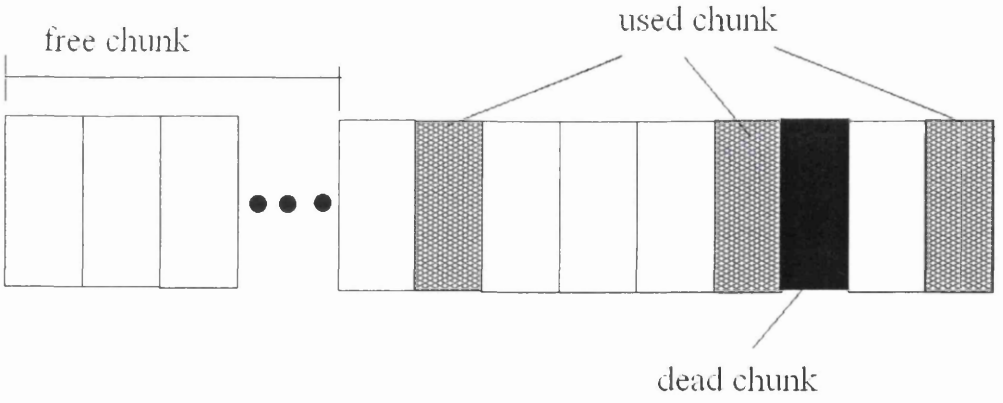


Figure 5.15: Schematic image of a memory pool

Algorithm 6 Data structure of a memory pool

```
struct mempool
{
void *blocks;                // list of allocated memory blocks
char *freechunk;             // next free chunk of memory
char *lastchunk;             // last chunk in current block
void *deadchunks;            // list of deallocated chunks of memory
unsigned int chunksize;      // size of a chunk
unsigned int chunksinblock; // number of memory chunks in a block
}
```

fragmentation because of variable memory block sizes, and their performance is poor. A more efficient solution is to pre-allocate a number of memory blocks with the same size called a *memory pool*, as shown in Fig. 5.15. Each rectangle represent a unit of the data structure. Usually a memory pool with 256, 512, 1024 or more units is pre-allocated at the beginning of the program. When a data set is created, it is put into a free chunk of the pool. When a data set is dismissed, the chunk of memory which it once occupied will be freed and available for new data sets. When a memory pool is used up, a new one with the same size (256, 512, 1024 or more) will be initiated. When the program ends, the whole memory pool will be freed. The memory pool can allocate, access and free blocks represented by pointers at runtime. Its data structure is shown in Algorithm 6.

Memory pools allow memory allocation with constant execution time with no fragmentation. The memory release for thousands of objects in a pool is just one operation, not thousands of operations. Memory pools also make it faster for the program to access large volume of data since they are all physically reside next to each other.

5.5.4 Sphere trees and implementation

In the dynamic BEM, the influence matrices $[H]^{nm}$ and $[G]^{nm}$ (n - current time step, m - past time step) have to be calculated at each time step according to the intersection between different size spheres and the boundary surface. In a computer, the shape of an object may be represented by tens of thousands of elements. In conventional BEM programs, the intersection check will loop over all triangles,

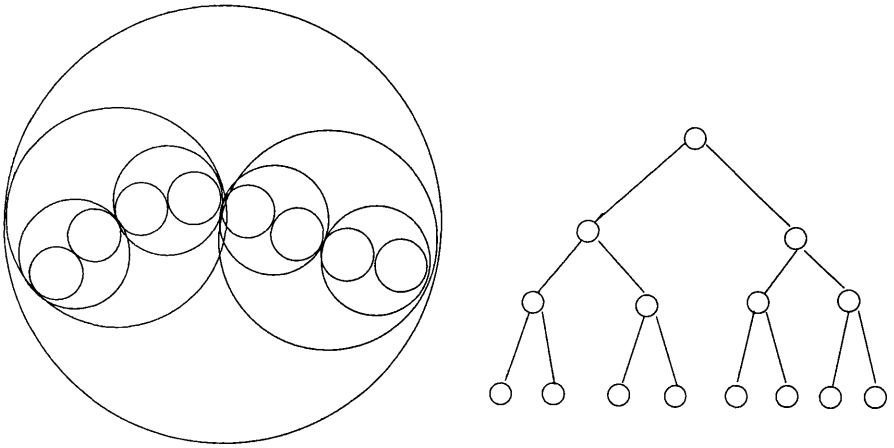


Figure 5.16: Schematic image of a sphere tree

Algorithm 7 Sphere tree data structure

```
struct sphtree
{
double center [3];      // sphere center
double radius;         // sphere radius
Surface_Triangle *tri; // stored triangle
unsigned char leaf;    // leaf flag
Sphere_Tree *lower;    // lower level nodes
Sphere_Tree *next;     // next sphere on this level
};
```

which requires a massive amount of computing time. In order to have a fast intersection search, triangles are replaced by the tree structure of hierarchical spheres which envelop them, as shown in Fig. 5.16. This reduces the amount of computing drastically and thus speeds up the solution. The data structure of sphere trees is shown as Algorithm 7. More algorithm details can be found in Section 5.2.4.

5.5.5 BEM integration modules

The basic BEM solution consist of computing the entries of the influence matrices G_{ij}, H_{ij} for each element and assembling them into the general matrices $[G]$ and $[H]$. The process is as follows :

Firstly, we loop over all vertexes on the surface mesh, using time step $n \quad (0..N-1)$ and step size dt to determine the two sphere's radii ($r_{min} = c \cdot ndt$, $r_{max} = c \cdot (n+1) \cdot dt$) which intersect with the boundary surface. The intersected sub-area

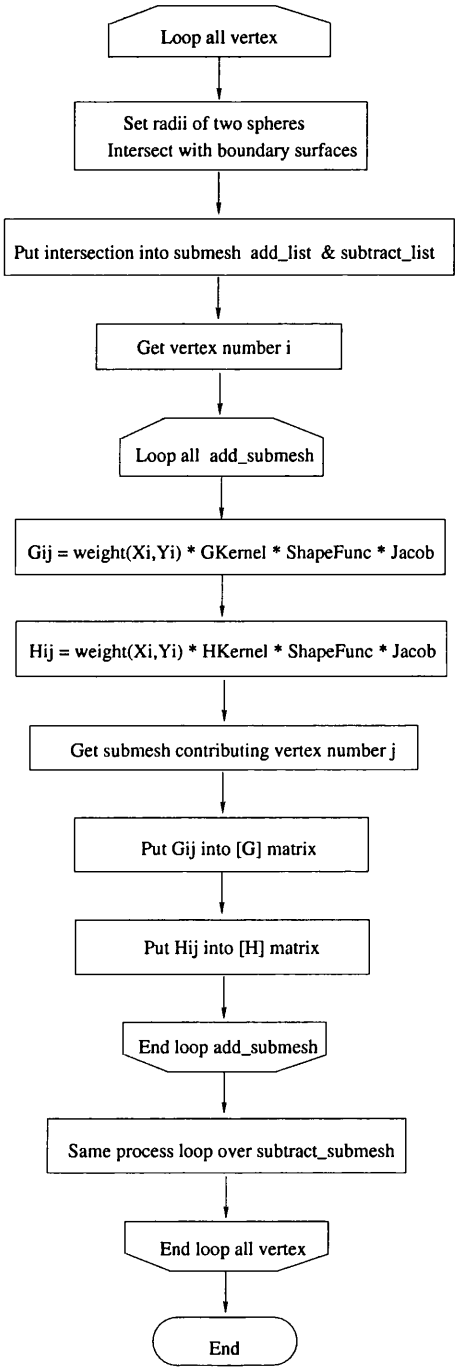


Figure 5.17: BEM integration process

of elements are put into the sub-mesh list of the vertex. The true intersection area is computed as the difference between the bigger sphere and smaller one: the add-list submesh means the intersection with the bigger sphere while the subtract-list submesh means the intersection with the smaller one. We loop over the submesh to numerically integrate G_{ij} and H_{ij} and assemble them into the general influence matrix $[G]$ and $[H]$. After looping over all vertexes in the mesh, all integrations for the integration representation formula are done.

5.5.6 Error estimation module

The algorithm for gradient-based error estimation for detecting the errors in each element is shown as Fig. 5.18. At each time step, we loop over all elements, compute the transformation matrix J according to Eq. 5.33; compute $\frac{\partial u}{\partial \xi}$, $\frac{\partial u}{\partial \eta}$, $\frac{\partial u}{\partial \tau}$ according to Eq. 5.32; perform the time integral of the terms analytically; then perform the space integral of the terms numerically. The same process will be applied to $\frac{\partial q}{\partial \xi}$, $\frac{\partial q}{\partial \eta}$, $\frac{\partial q}{\partial \tau}$. After looping over all elements, we can find the largest element error indicator of u and q. Then we loop over all elements again to mark those elements whose error indicators are larger than $tol \cdot max\,(e_u)$ or $tol \cdot max\,(e_q)$.

Following a similar process, the algorithm of re-solution error estimation is shown in Fig. 5.19.

5.5.7 Mesh refinement module

After the elements with large errors are found through error estimation methods, these triangular elements are refined further. The refinement strategy for the triangular mesh in this program is to divide the original triangle into two based on LEPP-refinement scheme. The algorithm is shown as in Fig. 5.20.

Firstly we loop over all elements to be refined; we find the LEPP triangle list of marked elements, and refine the last terminal pair in the LEPP triangle list. Then we create new vertexes with new coordinates, number etc.; put them into the vertex list of the new mesh as well as the relevant old vertexes. We also put new refined triangle elements into add_mesh list, and put old triangles into the delete_mesh

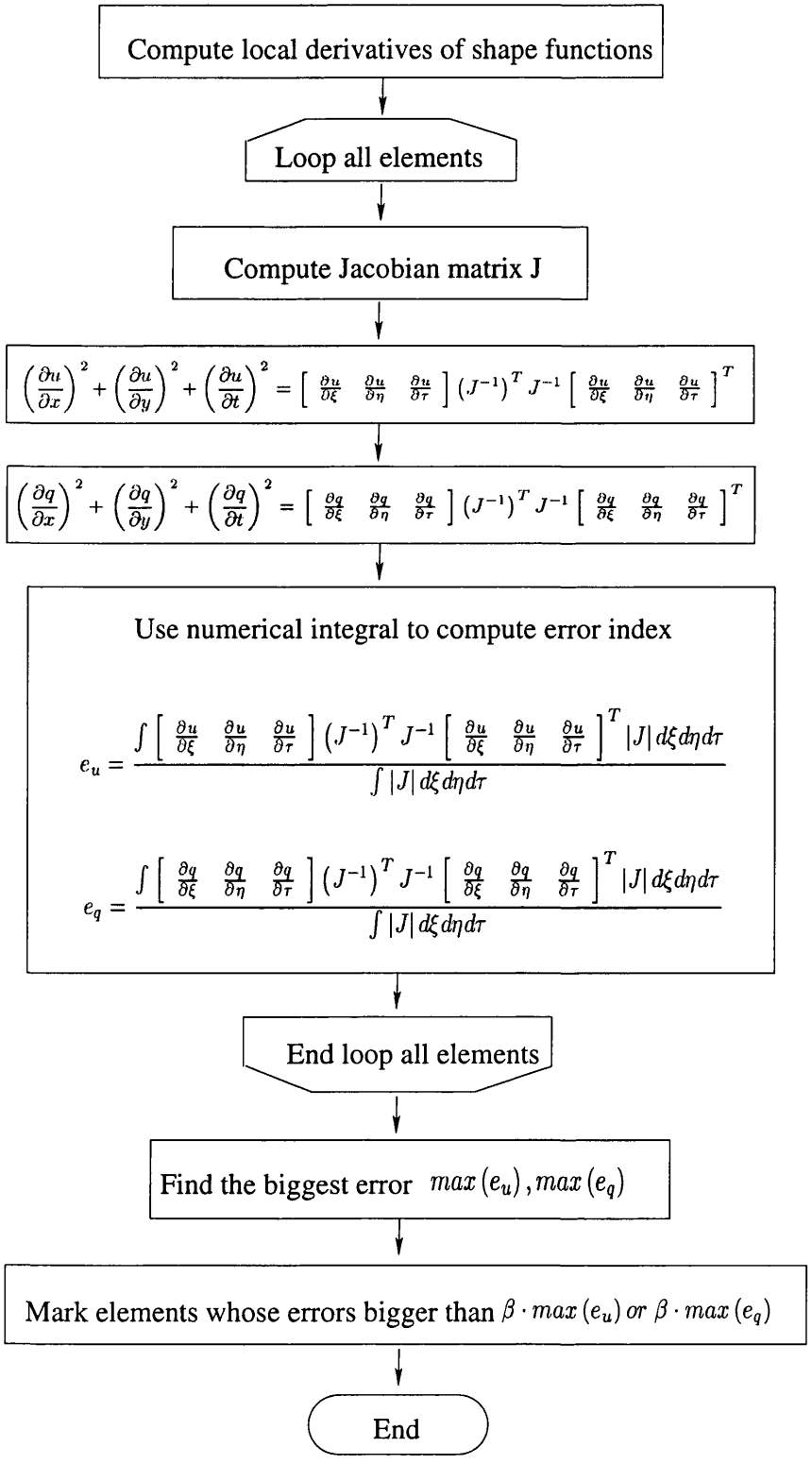


Figure 5.18: Gradient-based error estimation

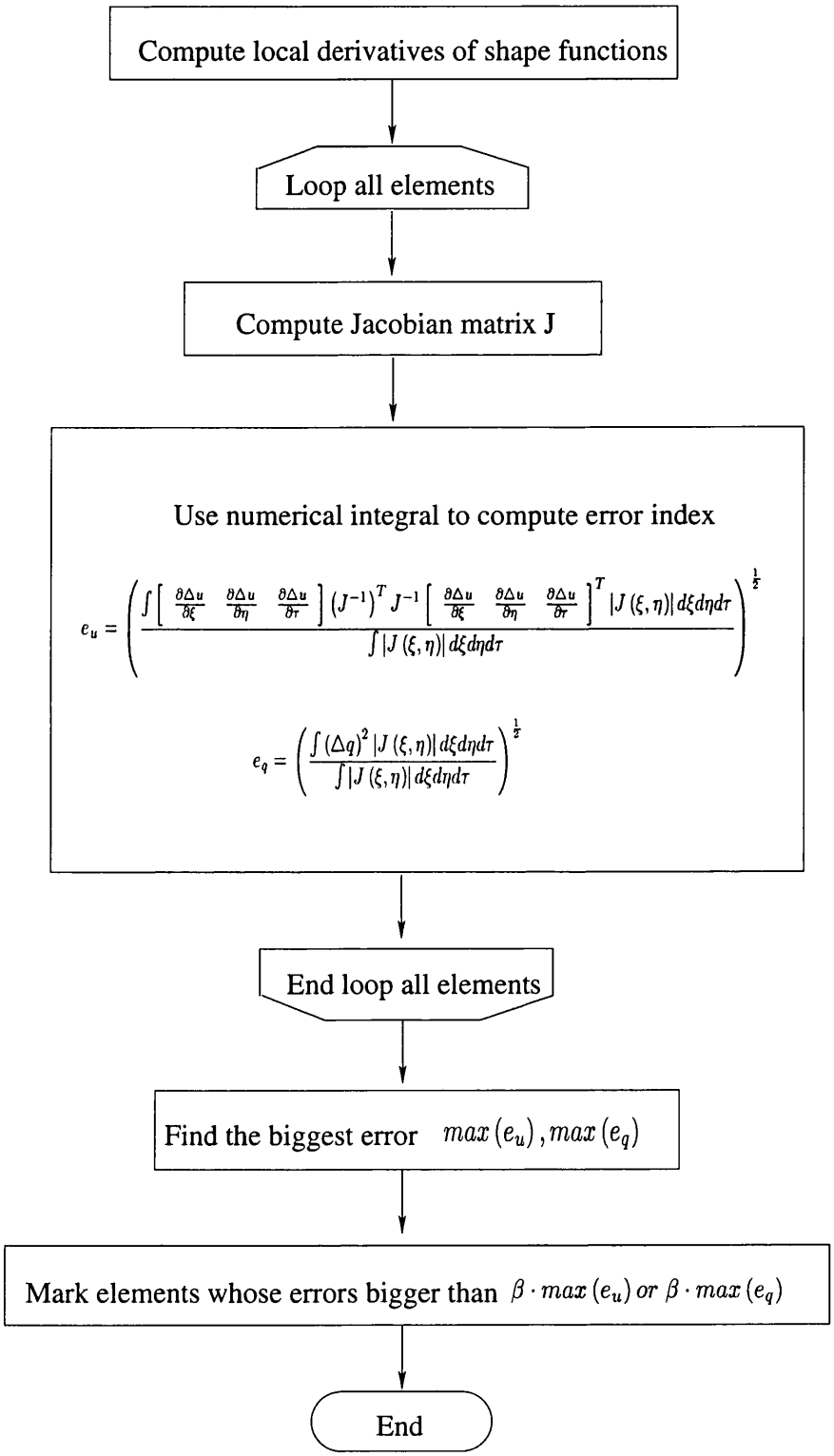


Figure 5.19: Re-solution error estimation

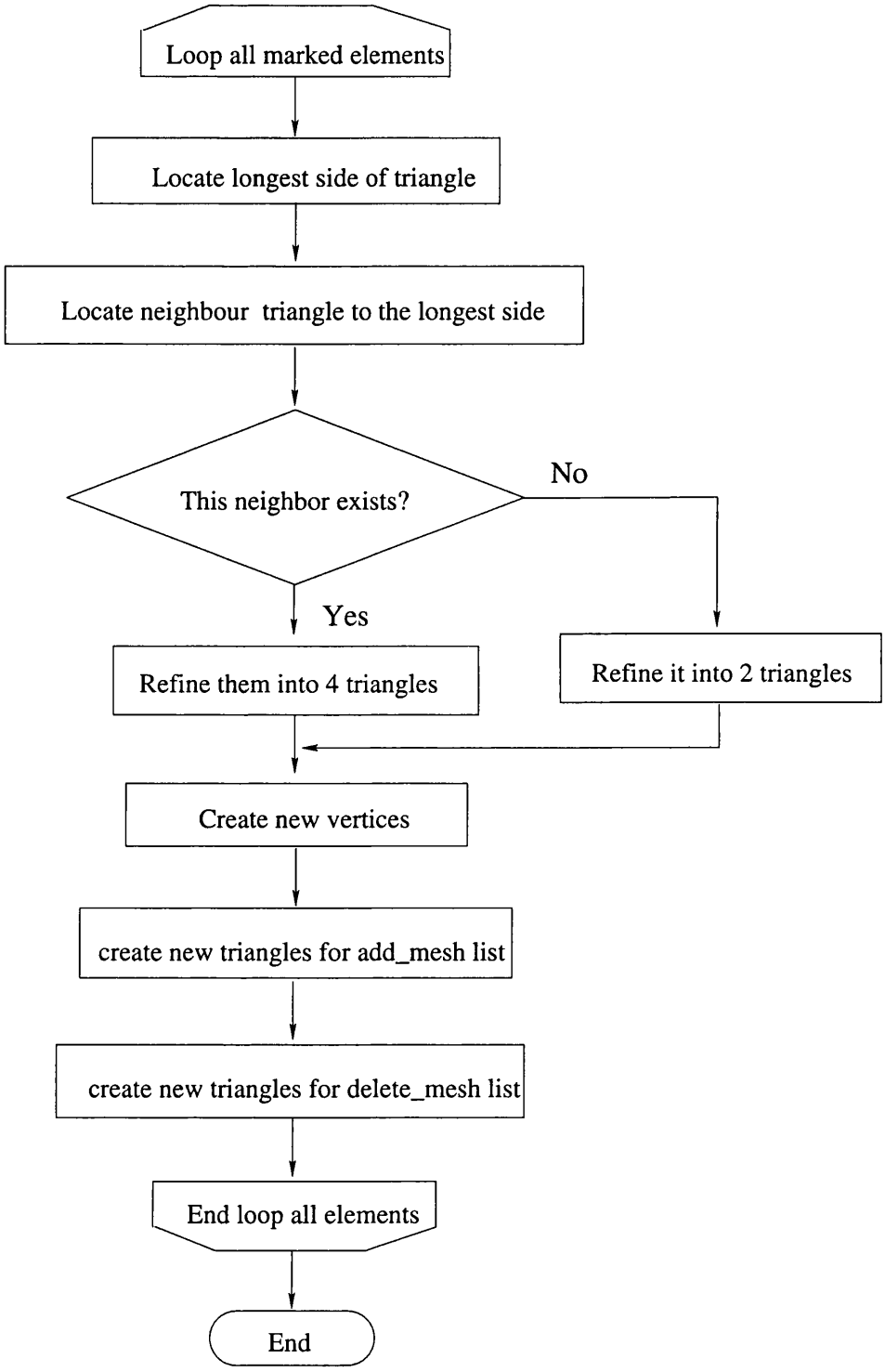


Figure 5.20: Algorithm of refining triangular meshes

list. The LEPP refinement continues until the marked element itself is refined. The same process is repeated for other marked triangles until all marked triangles are done.

5.5.8 Adaptive solver

Because the surface mesh is refined at different places at each time step, the conventional uniform mesh BEM solver is no longer adequate. Here the author developed a new dynamic BEM solver for adaptive meshes. The new solver is based on the observation that since the impulse wave fronts only occupy a small position in the total space-time, the refined elements are only small percentage of the whole mesh. Rather than creating a new mesh at each time step, we store one copy of the original mesh in the memory, then at each time step we only create a list of elements which are to be removed (called delete_mesh), and a list of elements which are to be added (add_mesh). The final mesh is the superposition of these three meshes, as shown in Fig. 5.21. Two large triangles are in the root mesh. If it needs to be refined, two large triangles will be put into delete_mesh, and four small triangles will be put into add_mesh.

The advantage of new algorithm is that coefficient matrices $[G]$ and $[H]$ of the original uniform mesh will not be re-computed. We only make changes to some entries based on delete_mesh, and augment the matrices to add new entries based on add_mesh, as shown in Fig. 5.22.

When computing coefficient matrices $[G^{nm}]$ and $[H^{nm}]$ from past influences, first we copy the matrices $[G^{nm}]$ and $[H^{nm}]$ of the uniform mesh into augmented matrices; then we loop over the vertexes in the root uniform mesh to intersect with delete_mesh and add_mesh at time step m, and put those entries into the upper right of the augmented matrices. If the mesh at time n itself is refined, we loop over the vertexes in add_mesh at the current time step n to intersect with the root uniform mesh and refined meshes (delete_mesh and add_mesh at time step m), and put those entries into the lower part of the augmented matrices. The algorithm is shown in Algorithm 8 below:

When computing coefficient matrices $[G^{nn}]$ and $[H^{nn}]$ at the current time step

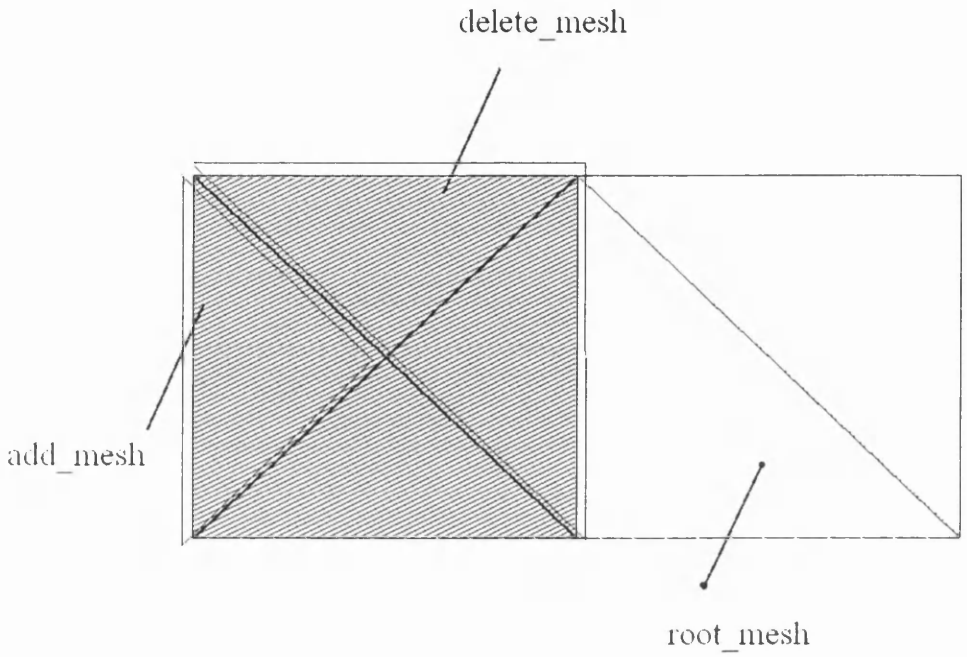


Figure 5.21: Superposition of adaptive meshes

$$\left[H^{nn}\right]\left\{u^{n}\right\}=\left[G^{nn}\right]\left\{P^{n}\right\}$$
$$+\sum_{m=1}^{n-1}\left[\left[H^{nm}\right]\left\{u^{m}\right\}+\left[G^{nm}\right]\left\{P^{m}\right\}\right]$$

Figure 5.22: BEM formula in the form of augmented matrices

Algorithm 8 Computing influence matrices from adaptive meshes

```
past_adaptive_mesh_influence_now(root_mesh,
                        adpMeshAdd,adpMeshDel,n,m)
1.  add_mesh = adpMeshAdd[m]
2.  del_mesh = adpMeshDel[m]
3.  copy root_mesh [G^nm],[H^nm]into augmented matrices
3.  if mesh at current time step n not refined
4.  then
5.    root_mesh intersect del_mesh, compute [G^nm],[H^nm]
5.    root_mesh intersect add_mesh, compute [G^nm],[H^nm]
6.  else
7.    add_mesh_now = adpMeshAdd[n]
5.    add_mesh_now intersect root_mesh,compute [G^nm],[H^nm]
5.    add_mesh_now intersect del_mesh, compute [G^nm],[H^nm]
5.    add_mesh_now intersect add_mesh, compute [G^nm],[H^nm]
8.  endif
9. end
```

n, first we copy the $[G^{nn}]$ and $[H^{nn}]$ matrices of the uniform mesh into augmented matrices; then we loop over the vertexes in the root uniform mesh to intersect with delete_mesh and add_mesh at the current time step n, and put those entries into the upper right of the augmented matrices. We also loop over the vertexes in add_mesh at time step n to intersect with the root uniform mesh, delete_mesh and add_mesh at time step n, and put those entries into the lower part of the augmented matrices. The algorithm is very similar to Algorithm 8.

5.6 Numerical examples

In this section, we present some numerical examples for 3D scalar wave propagation problems. As well as benchmark tests, we also demonstrate a reasonably realistic 3D application of an acoustic problem inside a car.

5.6.1 Wave propagation in a 3D bar

Wave propagation in a 3D bar with a uniform cross section, subjected to Heaviside loading, is shown in Fig. 5.23. This problem has been used as a benchmark test by several researchers (Mansur [54]) due to its simple geometry and its tendency

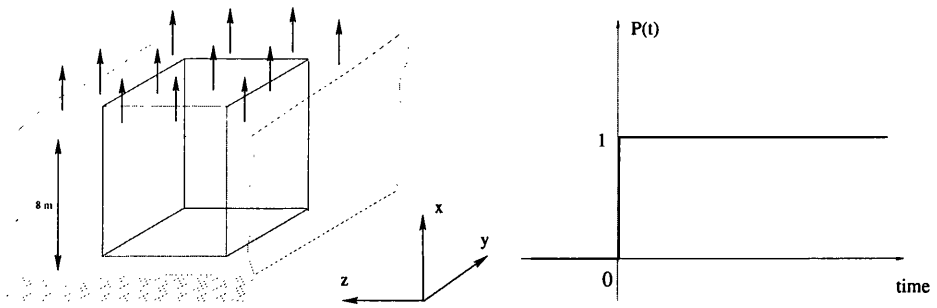


Figure 5.23: A 3D bar with a Heaviside load

to produce severe numerical instabilities. It can also be analyzed using a scalar formulation. It is essentially an one-dimensional problem because of its symmetry about two axes.

The bar's dimensions are $8m \times 4m \times 4m$. The uniform surface pressure $P(t)$ is a Heaviside function applied at the free end at $x = 8m$. The opposite is fixed where $x = 0m$. All other surfaces are traction-free. The wave velocity in the material is: $c = 200m/s$.

The boundary mesh consists of 320 quadratic triangular elements, as shown in Fig. 5.24. The temporal shape function is linear for displacements and constant for tractions. The time step size was defined by dimensionless parameter $\beta = c\Delta t/\Delta h$, where c is the wave velocity and Δh is equal to the radius of the circumscribed circle of the triangular element. During the BEM solution and mesh refinement, β keeps the value of 1.0 to keep the solution stable. In Fig. 5.24, the displacements on the boundary at various times are shown.

The computed time history of the normal traction at the fixed end, with various parameters $\beta = 0.3$ up to 1.4, is shown in Fig. 5.25, and is compared with the analytical solution. The x axis is the dimensionless time $\frac{ct}{L}$, and the y axis is the normalized traction $\frac{E \cdot P}{Pa}$. Stable results were obtained for $\beta > 0.5$; numerical damping increases as β increases.

The refinement process is invoked as the wave fronts move between the free surface and the fixed end. The computed time history of the normal traction at the fixed end, with various parameters $\beta = 0.3$ up to 1.4, is shown in Fig. 5.26. It is observed that the refinement scheme improves the accuracy and the stability for almost all cases except when $\beta = 0.3$. In this case, because the time step is too small

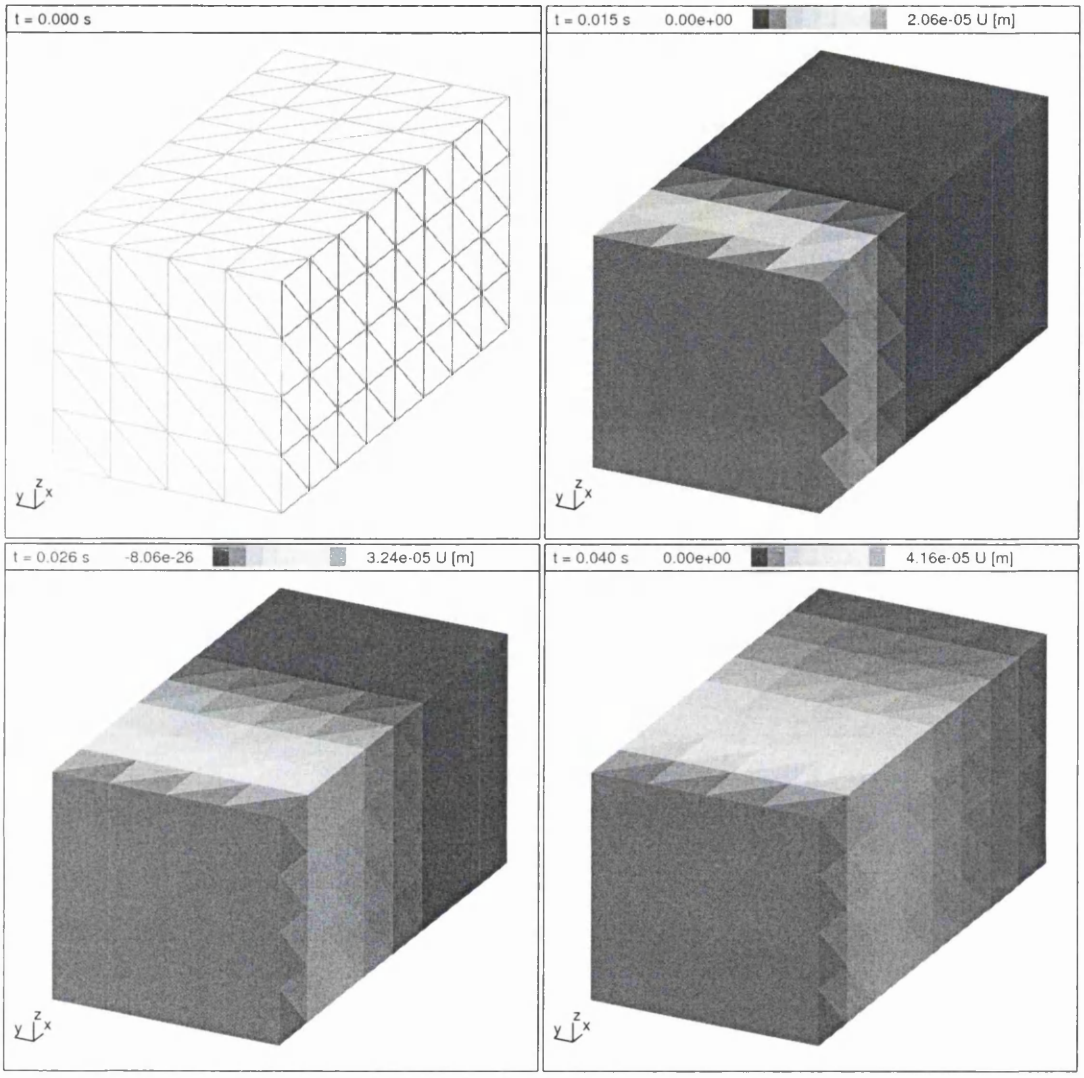


Figure 5.24: A 3D bar represented by 320 elements. Displacements at time t =0.015s, 0.026s, 0.040s

for the refinement scheme to be stable, the refinement scheme actually accelerates the instability.

While the geometric and boundary conditions are kept the same, an impulse load was applied in the free surface, defined as:

$$F(t) = e^{-\frac{(t-0.02)^2}{(0.01)^2}} \tag{5.41}$$

The time history of the traction at the fixed end is shown for both BEM without refinement and with refinement (Fig. 5.27). This figure clearly shows that the adaptive BEM solver has less artificial damping, less phase change, and remains

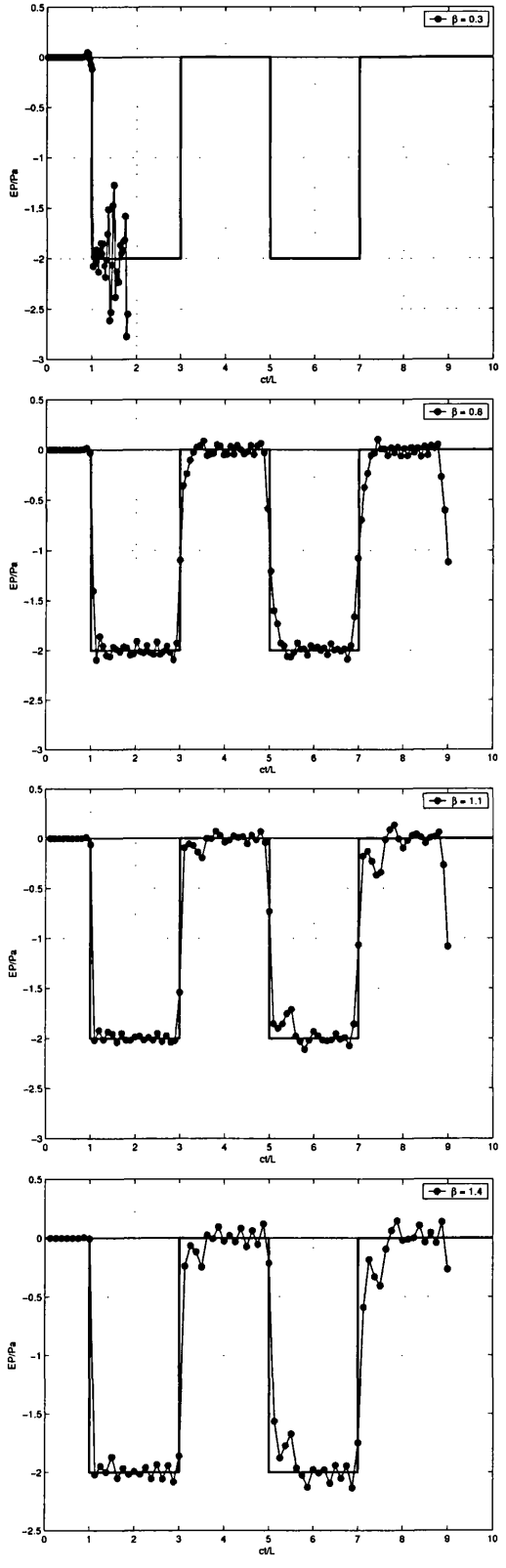


Figure 5.25: History of the normal traction at the fixed end for various β, using normal BEM solver

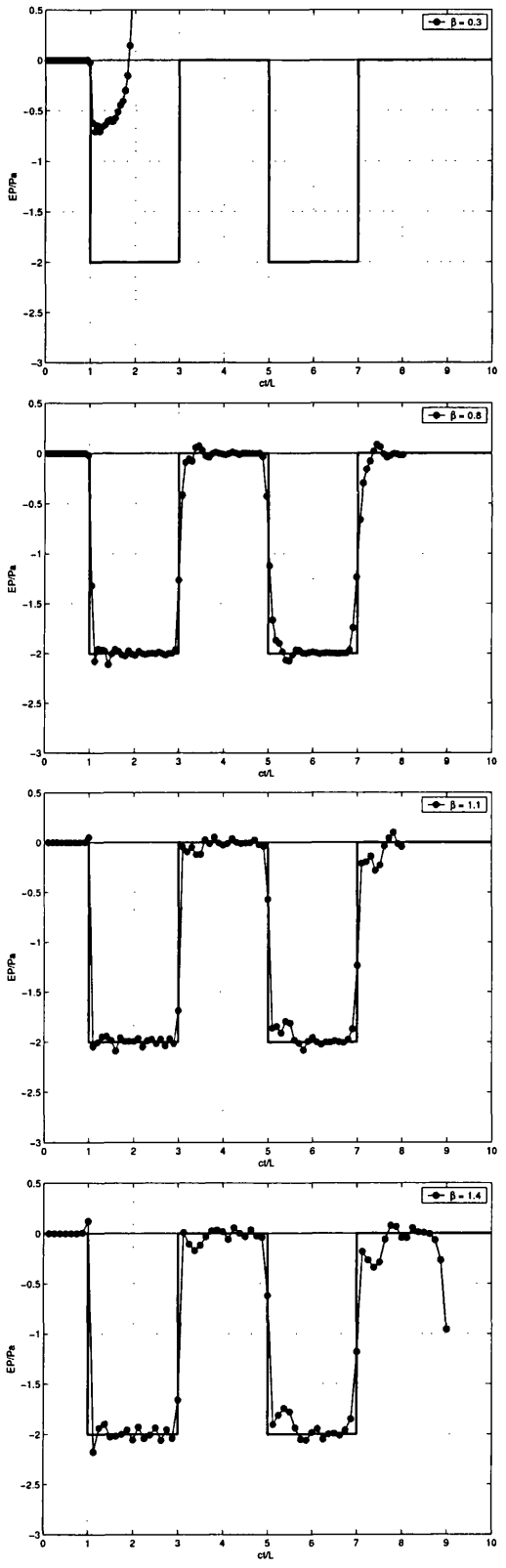
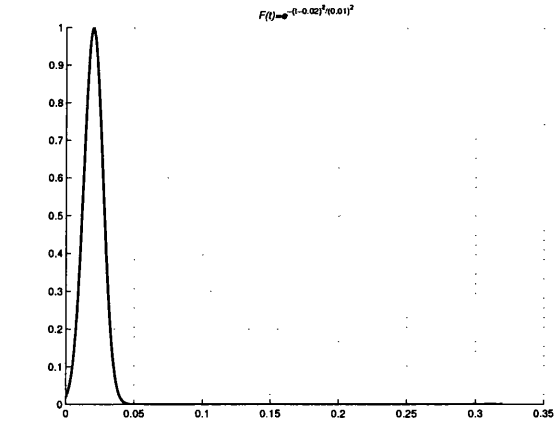
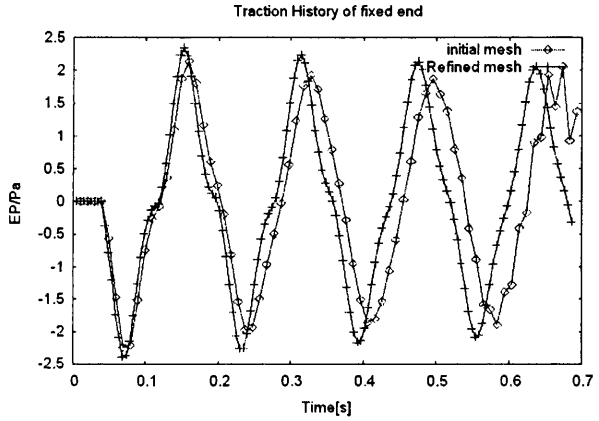


Figure 5.26: History of the normal traction at the fixed end for various different β, using adaptive BEM solver

stable while the non-adaptive approach fails.



(a)



(b)

Figure 5.27: (a) Time history of load; (b) Traction on the fixed end, with & without refined mesh

To obtain the efficiency of adaptive BEM, we record the computing time for each simulation with the time-space ratio β from 0.3 to 2.0. (Table 5.1). For the adaptive BEM, the sizes of the elements change at each time step. Thus, the final maximum time-space ratio β is used for the adaptive case. When β =1.1, the adaptive BEM is 41% faster than the conventional BEM. When the β values increase (that is, the initial mesh grows more coarse), the computing efficiency of adaptive BEM increases from 30% to 44%.

	$\beta = 0.3$	$\beta = 0.8$	$\beta = 1.1$	$\beta = 1.4$	$\beta = 2.0$
Conventional BEM (s)	N/A	406.2	234.5	186.0	100.7
adaptive BEM (s)	N/A	290.1	138.0	116.3	56.0

Table 5.1: The computing time of the 3D bar simulations with various β values

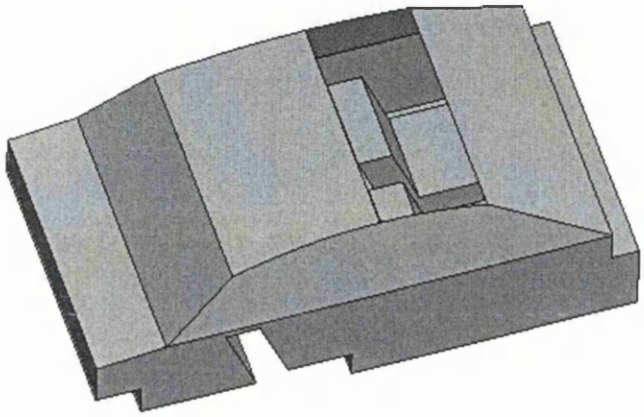


Figure 5.28: A model of interior acoustic field of a car

5.6.2 Acoustic field inside a car

A practical application of the adaptive dynamic BEM for the analysis of the interior acoustic field in a car is shown in Fig. 5.28, which is a crude model of the 3m long, 1.2m high and 1.8m wide car. The geometric model of the car was developed by Brooks and Morgans [9], which they analyzed using frequency-domain BEM. Humans can hear sounds with frequencies varied between $20 \sim 20,000$ Hz. To find out how people perceive sound at different places inside a car, we compute the acoustic field due to low-frequency cyclic noise from the engine of the car. The objectives of this simulation are to determine

- the acoustic pressure distribution inside the car for various frequencies;
- the locations of maximum acoustic pressure inside a car;
- the influence on stability and accuracy of the algorithm, for various element sizes, time-space ratios β, etc.

This problem is analyzed using a 3D scalar BEM formulation because it can be treated as a potential problem, when sound pressure p is equivalent to displacement u

and velocity v to traction t. (Wu, [88]). Here the sound velocity in air is $c = 343m/s$. Several test plans are designed as follows:

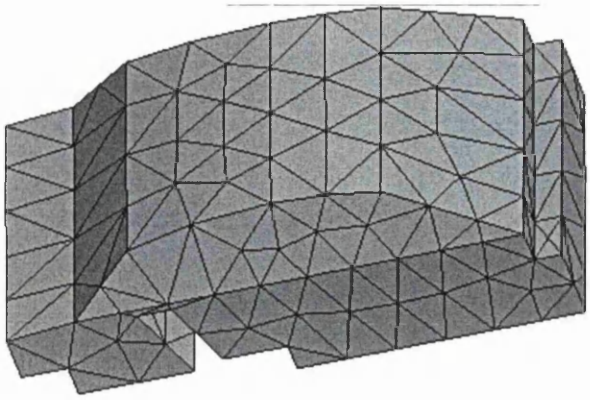
(a) The *pre-processor Gid* [29] is used to mesh the surface of the car; the mesh of 494 surface triangle elements and 974 nodes is shown in Fig. 5.29. The time-space ratio $\beta = 1.0$. A 100 Hz harmonic excitation representing the sound transmission through the engine firewall is applied to the vertical surface at the front of the car. The acoustic pressure distribution inside the car is computed, and the time history of the point with maximum acoustic pressure is drawn.

(b) A mesh with smaller elements (comprising 1,130 surface triangle elements and 2,364 nodes shown in Fig. 5.30), with a 440 Hz harmonic excitation (which is equivalent to the note middle C in music), is used to compute the acoustic pressure distribution again. The place and the absolute value of the maximum acoustic pressure is found and compared with case (a).

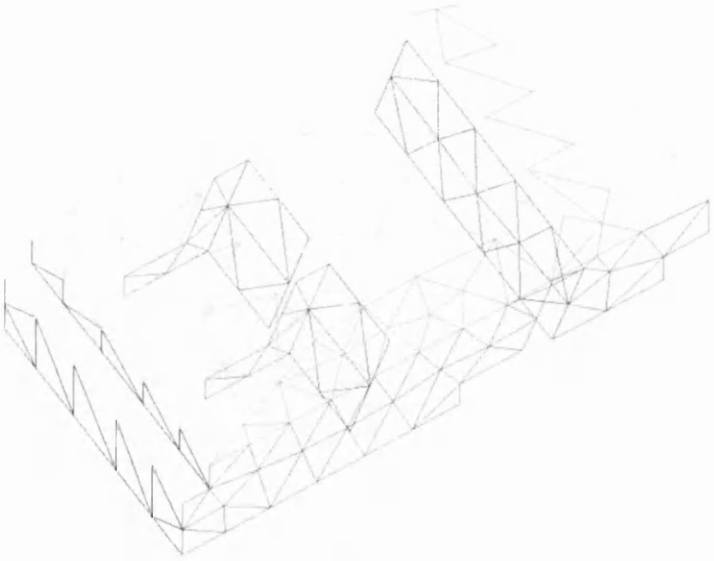
(c) A series of time-space ratios $\beta = 0.3 - 2.0$ are used to find its influence upon stability and accuracy.

In order to obtain good accuracy for harmonic wave propagation, at least four quadratic elements are needed per wave length. At $f = 100Hz$, the wave length $\lambda = 3.43m$. Since there are 15 elements per wave length in longitude, 28 elements per wave length in width, and 11 elements per wave length in height, the accuracy criterion is therefore satisfied for the mesh in case (a). Even in case (b), $f = 440Hz$, which yields a wave length $\lambda = 0.78m$, the criteria is also satisfied.

Because the radius of the circumscribed circle of the triangular elements of the car is between $r = 0.3 \sim 0.8m$, the proper time step is: $\triangle t = 0.0025 \sim 0.005s$. For acoustic wave $f = 100Hz$, after 100 time steps (roughly 5 periods of the harmonic load), the acoustic field achieves its steady state and the contour of the pressure field on the surface of the car is shown in Fig. 5.31. Not surprisingly, it is observed that the most noisy place is the front space of the car near the driver. However, it is less obvious that the corner of the back seats should also be quite noisy. Some sound absorbent material could be installed in both areas to improve the traveling experience.



(*a*)



(*b*)

Figure 5.29: (a) Car exterior meshes with 494 elements; (b) car interior mesh

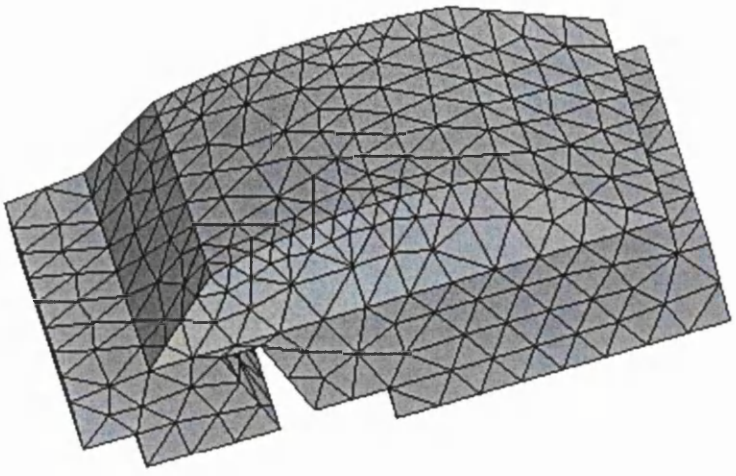


Figure 5.30: Car exterior surface meshes with 1,130 elements

Figure 5.31: Pressure magnitude inside the car, f = 100 Hz

For acoustic wave $f = 440Hz$, the radius of the circumscribed circle of the triangular elements of the car is between $r = 0.1 \sim 0.3m$; the proper time step is: $\triangle t = 0.0005 \sim 0.001s$. After the acoustic field achieves its steady state, the contour of the pressure field on the surface of the car is shown in Fig. 5.32. it is clear that the distribution of the maximum acoustic pressure for $f = 440Hz$ is different from the above one. The most noisy place is the front space of the car near the driver and the upper corner of the back seats. Therefore, the further study is needed to decide which noise frequency is dominant in the real driving situation. Some sound absorbent material can be placed in the right places accordingly.

For acoustic wave $f = 100Hz$, we select the time-space ratio β from 0.3 to 2.0. Numerical results show that it will not converge when $\beta \leq 0.3$ for conventional BEM (Table 5.2). For the adaptive BEM, the sizes of the elements change at each time step. Thus, the final maximum time-space ratio β is used for the adaptive case. When the β values increases, the conventional BEM becomes stable but lose accuracy by numerical damping while the adaptive BEM can still retain accuracy by refinement.

Figure 5.32: Pressure magnitude inside the car, f = 440 Hz

	$\beta = 0.3$	$\beta = 0.5$	$\beta = 0.8$	$\beta = 1.0$	$\beta = 2.0$
Conventional BEM	unstable	stable	stable	stable	stable
adaptive BEM	stable	stable	stable	stable	stable

Table 5.2: The stability of the numerical results with various β values

For acoustic wave $f = 100Hz$, we record the computing time for each simulation with the time-space ratio β from 0.3 to 2.0. (Table 5.3). When the β values decreases, the computing time increases greatly. To reach the similar accuracy, the adaptive BEM is 37.5% faster than the conventional BEM.

Several conclusions may be drawn from this numerical examples:

- For the acoustic wave $f = 100Hz$, the maximum sound pressure locates in the front of a car near the engine and the corner of the back seats.
- When the acoustic wave has a different frequency, the distribution of the maximum sound pressure changes.

	$\beta = 0.3$	$\beta = 0.5$	$\beta = 0.8$	$\beta = 1.0$	$\beta = 2.0$
Conventional BEM (s)	N/A	2907.2	1711.2	1362.5	707.9
adaptive BEM (s)	2898.4	1817.0	1069.5	851.6	442.5

Table 5.3: The computing time of the simulations with various β values

- When the smaller β are used, the conventional BEM algorithm loses its stability while the adaptive BEM is still stable.
- To reach the similar accuracy, the adaptive BEM is about $30\% \sim 40\%$ faster than the conventional BEM.

5.7 Summary

Based on the BEM integral equation for 3D transient acoustic wave problems, the boundary domain is discretized into 6-node curved triangles. The linear system equations are assembled from the element influence matrices which are computed from the space-time integrals of Green's functions. In order to better approximate high gradient areas, gradient-based and resolution-based error indicators are used to locate them, and space-refinement schemes and local time stepping are employed to better resolve the moving wavefront, while retaining computational efficiency. We apply this method to solve the problems of wave propagation in a 3D bar and show that, compared with traditional dynamic BEM, the solution is more accurate, less artificially-damped and more stable. Finally, we analyze the acoustic field inside a car, and demonstrate that a reasonably accurate simulation for this complex problem can be obtained using an ordinary desktop PC.

Chapter 6

Adaptive Boundary Element Method for elastodynamics in 3D

6.1 Introduction

In principle, for many complex dynamic problems in geomechanics such as earthquake wave propagation, soil-structure interaction etc., elastodynamic BEM in 3D is a powerful numerical method.

In this chapter, the numerical implementation of the boundary integral equations for 3D elastodynamics is described. In Section 6.2, we discuss the discretization of BEM formulation of the 3D elastodynamics vector field equation; the evaluation of the time integral, and singular and non-singular kernel integrations. In Section 6.3, we introduce error estimates based on stress recovery and the consequent adaptive schemes for 3D elastodynamics and local time stepping. In Section 6.4, various aspects of BEM programming, such as the vector field kernel time integrations, and stress resolution based error estimation etc., are discussed in detail. Finally, some examples of the application of adaptive space-time BEM are given and discussed in Section 6.5.

6.2 Space-time BEM for elastodynamics in 3D

6.2.1 Boundary integral formulation for space-time BEM elastodynamics

The governing equations of elastodynamics are:

$$(c_1^2 - c_2^2)\mathbf{u}_{j,ij} + c_2^2\mathbf{u}_{i,jj} + \mathbf{f}_i = \ddot{\mathbf{u}}_i \tag{6.1}$$

where $\mathbf{u}_i(\mathbf{x}, t)$ is the displacement in 4D space-time in the *ith* direction ($i = 1, 2, 3$) and $\ddot{\mathbf{u}}_i$ is the corresponding acceleration. $\mathbf{x}$ denotes the spatial coordinates (x, y, z); $\mathbf{f}_i = \mathbf{b}_i/\rho$, $\mathbf{b}_i$ is the body force; ρ is the density; c_1 is the pressure wave speed and c_2 is the shear wave speed.

The displacements are specified on S_1, while tractions are specified on S_2:

$$\mathbf{u}_i(\mathbf{x}, t)\,|_{S_1} = \overline{\mathbf{u}}_i(\mathbf{x}, t) \qquad \mathbf{p}_i(\mathbf{x}, t)\,|_{S_2} = \overline{\mathbf{p}}_i(\mathbf{x}, t) \tag{6.2}$$

where the boundary $S_1 \cup S_2 = S$, $S_1 \cap S_2 = \emptyset$.

Initial conditions are prescribed at time $t = 0$:

$$\mathbf{u}_i(\mathbf{x}, 0)\,|_S = \mathbf{u}_{0i}(\mathbf{x}) \qquad \dot{\mathbf{u}}_i(\mathbf{x}, 0)\,|_S = \mathbf{v}_{0i}(\mathbf{x}) \tag{6.3}$$

$\mathbf{u}_{0i}(\mathbf{x})$ are the initial displacements on the boundary, and $\mathbf{v}_{0i}(\mathbf{x})$ are the initial velocities on the boundary. Assuming zero body forces $\mathbf{f}_i = 0$, and quiescent initial conditions $\mathbf{u}_{0i}(\mathbf{x}) = 0$ and $\mathbf{v}_{0i}(\mathbf{x}) = 0$, the 3D elastodynamic BEM integral equation is: (Dominguez [20])

$$\begin{aligned} c_{lk}^i u_k^i(\boldsymbol{x}^i, t) = &\int_S u_{lk}^*(\boldsymbol{x}, t-\tau; \boldsymbol{x}^i) * p_k(\boldsymbol{x}, \tau)\, dS(\boldsymbol{x}, \tau) \\ &- \int_S p_{lk}^*(\boldsymbol{x}, t-\tau; \boldsymbol{x}^i) * u_k(\boldsymbol{x}, \tau) dS(\boldsymbol{x}, \tau) \end{aligned} \tag{6.4}$$

where, $\boldsymbol{x}^i$ refers to a specific collocation point on boundary S. The Green's function u_{lk}^* is [20]:

$$
\begin{aligned}
u^*_{lk}\left(\boldsymbol{x},t;\boldsymbol{x}^i\right) =& \frac{1}{4\pi\rho}\{\ \frac{t}{r^2}\left(\frac{3r_{,l}r_{,k}}{r}-\frac{\delta_{lk}}{r}\right) \\
& [H\left(t-\frac{r}{c_1}\right)-H\left(t-\frac{r}{c_2}\right)]\} \\
& +\frac{1}{4\pi\rho}\{\ \frac{r_{,l}r_{,k}}{r}[\frac{1}{c_1^2}\delta(t-\frac{r}{c_1})-\frac{1}{c_2^2}\delta(t-\frac{r}{c_2})] \\
& +\frac{\delta_{lk}}{rc_2^2}\delta(t-\frac{r}{c_2})\}
\end{aligned}
\tag{6.5}
$$

where $r = |\boldsymbol{x}-\boldsymbol{x}^i|$, $H(x)$ is the Heaviside function and $\delta(x)$ is the Dirac-delta function. The traction kernel function $p^*_{lk}\left(\boldsymbol{x},t;\boldsymbol{x}^i\right)$ can be obtained from the Green's function u^*_{lk} , using Hooke's law and the equilibrium equation, which yields:

$$
\begin{aligned}
p^*_{lk}\left(\boldsymbol{x},t;\boldsymbol{x}^i\right) = \frac{1}{4\pi}\{\ & \left(\frac{\partial r}{\partial n}\delta_{lk}+r_{,k}n_l\right)A \\
& + r_{,l}r_{,k}\frac{\partial r}{\partial n}B + r_{,l}n_kC\}
\end{aligned}
\tag{6.6}
$$

where the coefficients A, B, C are :

$$
\begin{aligned}
A &= \frac{6c_2^2}{r^4}t\,[H(t-\frac{r}{c_1})-H(t-\frac{r}{c_2})]-\frac{2}{r^2}[[\delta(t-\frac{r}{c_2})-\frac{c_2^2}{c_1^2}\delta(t-\frac{r}{c_1})] \\
&- \frac{1}{r^2}[\delta(t-\frac{r}{c_2})-\frac{r}{c_2}\dot{\delta}(t-\frac{r}{c_2})] \\
B &= \frac{-30c_2^2}{r^4}t\,[H(t-\frac{r}{c_1})-H(t-\frac{r}{c_2})] \\
&+ \frac{12}{r^2}[[\delta(t-\frac{r}{c_2})-\frac{c_2^2}{c_1^2}\delta(t-\frac{r}{c_1})] \\
C &= \frac{6c_2^2}{r^4}t\,[H(t-\frac{r}{c_1})-H(t-\frac{r}{c_2})] \\
&- \frac{2}{r^2}[[\delta(t-\frac{r}{c_2})-\frac{c_2^2}{c_1^2}\delta(t-\frac{r}{c_1})]
\end{aligned}
\tag{6.7}
$$

In Eq. (6.4), the integrals of the products of $p_k(\boldsymbol{x},\tau)$ and the Heaviside function $H\left(t-\frac{r}{c}\right)$ or the Dirac-delta function $\delta\left(t-\frac{r}{c}\right)$ in space-time become spatial integrals of $p_k(\boldsymbol{x},\tau)$ over the boundary. Most of the integrals of the product of $u_k(\boldsymbol{x},\tau)$ and the fundamental traction solution can be similarly reduced to spatial integrals. For

example, the product of the integral of the displacement $u_k(\boldsymbol{x},\tau)$ and the time derivative of Dirac delta function is:

$$\int \frac{\partial \delta(t-\frac{r}{c_1})}{\partial \tau} u_k(\boldsymbol{x},\tau) dS(\boldsymbol{x},\tau) = \int \frac{\partial u_k(\boldsymbol{x},t-\frac{r}{c_1})}{\partial \tau} dS(\boldsymbol{x}) \tag{6.8}$$

Here, we introduce a new interpolation scheme. Not only is the spatial domain interpolated via shape functions, but also both temporal domains are interpolated using $N^k(\xi,\eta,\tau)$ as a space-time interpolation function. Thus:

$$\begin{aligned} u_k &= \sum_j N^k(\xi,\eta,\tau)\, u_k^j; \qquad p_k = \sum_j N^k(\xi,\eta,\tau)\, p_k^j; \\ t_k &= \sum_j N^k(\xi,\eta,\tau)\, t_k^j; \end{aligned} \tag{6.9}$$

where u_k^j, p_k^j and t_k^j are nodal values of displacement, traction and time respectively.

Distance in space-time is defined as:

$$|ct-r| = \left| c(t_i - \sum N^k(\xi,\eta,\tau)\cdot t_j^k) - r \right| \tag{6.10}$$

where, t_i is the time of the source point $\mathbf{x}_i$, t_j^k is the time of the field point in element j, r is the distance between the source point and the field point, and c is the wave speed.

After this numerical approximation, Eq. (6.4) takes the form:

$$\begin{aligned} c_{lk}^i u_k^i = \sum_{j=1}^{N} (& \int_{S_j} u_{lk}^* \cdot N^j(\xi,\eta,\tau)\, p_k^j dS(\boldsymbol{x},\tau) \\ & - \int_{S_j} p_{lk}^* \cdot N^j(\xi,\eta,\tau)\, u_k^j dS(\boldsymbol{x},\tau)) \end{aligned} \tag{6.11}$$

where u_k^j, p_k^j are nodal values of displacement and traction, $N^j(\xi,\eta,\tau)$ are shape functions, S_j is the surface boundary element, and N is the total number of boundary elements.

Letting

$$\begin{aligned} G_{lk}^{ij} &= \int_{S_j} u_{lk}^{*} \cdot N^j(\xi,\eta,\tau)\, dS(\boldsymbol{x},\tau) \\ \hat{H}_{lk}^{ij} &= \int_{S_j} p_{lk}^{*} \cdot N^j(\xi,\eta,\tau)\, dS(\boldsymbol{x},\tau) \end{aligned} \tag{6.12}$$

we adopt the notation:

$$H_{lk}^{ij} = \begin{cases} \hat{H}^{ij} & i \neq j \\ \hat{H}^{ij} + c_{lk}^{i} & i = j \end{cases} \tag{6.13}$$

The system of equations for all boundary nodes can be expressed in the matrix form:

$$\sum_{j=1}^{N} H_{lk}^{ij} u_k^j = \sum_{j=1}^{N} G_{lk}^{ij} p_k^j \tag{6.14}$$

where N is the total number of nodes in the space-time domain.

The causality law requires that boundary values at later time are only influenced by quantities at earlier time, but not vice versa. Thus, numerical methods constructed from these space-time boundary integral equations are global in time, i.e., it is necessary to compute the solution for all time steps from the beginning to obtain the current solution. The system is solved step-by-step: once $\mathbf{u}$ and $\mathbf{p}$ are known for the previous time steps, the solution for the *nth* time step is obtained from:

$$\begin{aligned} \mathbf{H}_{lk}^{(nn)ij}\mathbf{u}_k^{(n)j} = \mathbf{G}_{lk}^{(nn)ij}\mathbf{p}_k^{(n)j} \\ + \sum_{m=1}^{n-1} \mathbf{G}_{lk}^{(nm)ij}\mathbf{p}_k^{(m)j} - \mathbf{H}_{lk}^{(nm)ij}\mathbf{u}_k^{(m)j} \end{aligned} \tag{6.15}$$

where n is the current time step, m is the past time steps, point i is the source point, point j is the field point. l is the *lth* component of the vector value on the source point; k is the *kth* component of the vector value on the field point. $\mathbf{H}$ and $\mathbf{G}$ are the influence matrices; $\mathbf{u}$and $\mathbf{p}$ are vector values of displacements and tractions. The boundary consists of the spatial boundary extruded into space-time, like that of cylinder extruded from a 2D circle into 3D space. The increase in dimensionality is offset by special features of the problem. Therefore, it remains the same dimensionality as static problems, as discussed in Section 6.2.2.

6.2.2 Evaluation of time kernel integral

Generally in the space-time scheme, the time kernel integral is treated by using numerical quadrature like the space integral. However, since the adaptive scheme will be introduced into a uniform mesh in space-time, the time integrals are evaluated analytically to maintain both efficiency and accuracy. In order to integrate the time convolution analytically, the time interval of interest t is discretized into N time steps of duration $\triangle t$, so that $t = n \cdot \triangle t$, where $n = 1, 2, ..., N$. The current time is denoted as τ. A linear time variation is assumed for displacements while a constant one is assumed for tractions at a given point x_i as shown in Fig. 6.1. The displacement is explicitly written as:

$$u(\mathbf{x}, t) = \sum_{m=1}^{N} \left[\frac{\tau - \tau_{m-1}}{\tau_m - \tau_{m-1}} u_m(\mathbf{x}) + \frac{\tau_m - \tau}{\tau_m - \tau_{m-1}} u_{m-1}(\mathbf{x}) \right] \Phi_m(t) \tag{6.16}$$

where u_m and u_{m-1} are spatial variation of the displacements at time τ_m and τ_{m-1} respectively. The function $\Phi_m(t)$ is a piece-wise linear shape function in time:

$$\Phi_m(t) = H\left[\tau - (m-1) \triangle t\right] - H\left[\tau - m \triangle t\right] \tag{6.17}$$

Tractions with constant time variation are written as:

$$P(\mathbf{x}, t) = \sum_{m=1}^{N} \left[P_m(\mathbf{x})\right] \Phi_m(t) \tag{6.18}$$

where P_m is the spatial variation of the traction at time τ_m.

To evaluate the time kernel integral of the displacement u_k in Eq. 6.4, we write down the Green's function u^*_{lk} in matrix form:

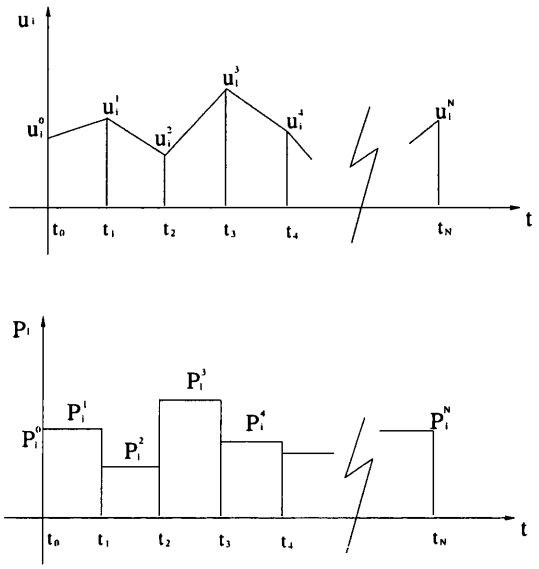


Figure 6.1: Linear variation of displacement u_i and constant variation of traction P_i over a time interval

$$
\begin{aligned}
&\begin{bmatrix} u^*_{11} & u^*_{12} & u^*_{13} \\ u^*_{21} & u^*_{22} & u^*_{23} \\ u^*_{31} & u^*_{32} & u^*_{33} \end{bmatrix} \\
= \; & \frac{t}{4\pi\rho r^3} \left\{ \left[3 \cdot \begin{bmatrix} \frac{\partial r}{\partial x}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial x}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial x}\frac{\partial r}{\partial z} \\ \frac{\partial r}{\partial y}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial y}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial y}\frac{\partial r}{\partial z} \\ \frac{\partial r}{\partial z}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial z}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial z}\frac{\partial r}{\partial z} \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right] \right\} \left[H\left(t - \frac{r}{c_1}\right) - H\left(t - \frac{r}{c_2}\right) \right] \\
+ \; & \frac{1}{4\pi\rho r} \begin{bmatrix} \frac{\partial r}{\partial x}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial x}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial x}\frac{\partial r}{\partial z} \\ \frac{\partial r}{\partial y}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial y}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial y}\frac{\partial r}{\partial z} \\ \frac{\partial r}{\partial z}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial z}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial z}\frac{\partial r}{\partial z} \end{bmatrix} \left[\frac{1}{c_1^2}\delta\left(t - \frac{r}{c_1}\right) - \frac{1}{c_2^2}\delta\left(t - \frac{r}{c_2}\right) \right] \\
+ \; & \frac{1}{4\pi\rho r c_2^2} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \delta\left(t - \frac{r}{c_2}\right)
\end{aligned}
\tag{6.19}
$$

Assuming that the shape function for tractions is constant, we put this matrix formula into the time integral $[\tau_{m-1}, \tau_m]$, and split the resulting integral into 3 parts:

$$
U^{nm} = \int_{\tau_{m-1}}^{\tau_m} u^*_{lk} \cdot 1 \cdot d\tau = I_1 + I_2 + I_3 \tag{6.20}
$$

The first term is written as follows:

$$I_1 = \int_{\tau_{m-1}}^{\tau_m} \left\{ \frac{(t-\tau)}{4\pi\rho r^3} \left[3 \cdot \begin{bmatrix} \frac{\partial r}{\partial x}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial x}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial x}\frac{\partial r}{\partial z} \\ \frac{\partial r}{\partial y}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial y}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial y}\frac{\partial r}{\partial z} \\ \frac{\partial r}{\partial z}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial z}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial z}\frac{\partial r}{\partial z} \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right] \cdot \left[H\left((t-\tau) - \frac{r}{c_1}\right) - H\left((t-\tau) - \frac{r}{c_2}\right) \right] \right\} d\tau \tag{6.21}$$

Eq. 6.21 means the effects upon point $\mathbf{x}^i$ at time t from the point $\mathbf{x}^j$ between times $\left[(t-\tau) - \frac{r}{c_1}\right]$ and $\left[(t-\tau) - \frac{r}{c_2}\right]$.

The second term is:

$$I_2 = \int_{\tau_{m-1}}^{\tau_m} \frac{1}{4\pi\rho r} \begin{bmatrix} \frac{\partial r}{\partial x}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial x}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial x}\frac{\partial r}{\partial z} \\ \frac{\partial r}{\partial y}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial y}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial y}\frac{\partial r}{\partial z} \\ \frac{\partial r}{\partial z}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial z}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial z}\frac{\partial r}{\partial z} \end{bmatrix} \left[\frac{1}{c_1^2}\delta\left(t - \frac{r}{c_1}\right) - \frac{1}{c_2^2}\delta\left(t - \frac{r}{c_2}\right) \right] P d\tau$$

$$= \frac{1}{4\pi\rho r} \begin{bmatrix} \frac{\partial r}{\partial x}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial x}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial x}\frac{\partial r}{\partial z} \\ \frac{\partial r}{\partial y}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial y}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial y}\frac{\partial r}{\partial z} \\ \frac{\partial r}{\partial z}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial z}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial z}\frac{\partial r}{\partial z} \end{bmatrix} \left[\frac{1}{c_1^2} P\Big|_{t-\frac{r}{c_1}} - \frac{1}{c_2^2} P\Big|_{t-\frac{r}{c_2}} \right] \tag{6.22}$$

This equation means that the current displacement $u(\mathbf{x}, t)$ is influenced by the term at early times defined by backward cones ($r_1 = c_1(t-\tau)$ & $r_2 = c_2(t-\tau)$) and the space boundaries.

The third term is:

$$I_3 = \int_{\tau_{m-1}}^{\tau_m^+} \frac{1}{4\pi\rho r c_2^2} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \delta\left(t - \frac{r}{c_2}\right) P d\tau$$

$$= \frac{1}{4\pi\rho r c_2^2} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \frac{1}{c_2^2} P\Big|_{t-\frac{r}{c_2}} \tag{6.23}$$

Here, the current displacement $u(\mathbf{x}, t)$ is influenced by the term at early times defined by the backward cone ($r_2 = c_2(t - \tau)$) and the space boundaries.

The traction p^*_{lk} is written in matrix form:

$$\begin{aligned}
&\begin{bmatrix} p^*_{11} & p^*_{12} & p^*_{13} \\ p^*_{21} & p^*_{22} & p^*_{23} \\ p^*_{31} & p^*_{32} & p^*_{33} \end{bmatrix} \\
&= \frac{1}{4\pi} \cdot A \cdot \left\{ \left[\frac{\partial r}{\partial n} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} \frac{\partial r}{\partial x}\frac{\partial n}{\partial x} & \frac{\partial r}{\partial x}\frac{\partial n}{\partial y} & \frac{\partial r}{\partial x}\frac{\partial n}{\partial z} \\ \frac{\partial r}{\partial y}\frac{\partial n}{\partial x} & \frac{\partial r}{\partial y}\frac{\partial n}{\partial y} & \frac{\partial r}{\partial y}\frac{\partial n}{\partial z} \\ \frac{\partial r}{\partial z}\frac{\partial n}{\partial x} & \frac{\partial r}{\partial z}\frac{\partial n}{\partial y} & \frac{\partial r}{\partial z}\frac{\partial n}{\partial z} \end{bmatrix} \right] \right\} \\
&+ \frac{1}{4\pi} \cdot B \cdot \frac{\partial r}{\partial n} \cdot \begin{bmatrix} \frac{\partial r}{\partial x}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial x}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial x}\frac{\partial r}{\partial z} \\ \frac{\partial r}{\partial y}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial y}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial y}\frac{\partial r}{\partial z} \\ \frac{\partial r}{\partial z}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial z}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial z}\frac{\partial r}{\partial z} \end{bmatrix} + \frac{1}{4\pi} \cdot C \cdot \begin{bmatrix} \frac{\partial r}{\partial x}\frac{\partial n}{\partial x} & \frac{\partial r}{\partial x}\frac{\partial n}{\partial y} & \frac{\partial r}{\partial x}\frac{\partial n}{\partial z} \\ \frac{\partial r}{\partial y}\frac{\partial n}{\partial x} & \frac{\partial r}{\partial y}\frac{\partial n}{\partial y} & \frac{\partial r}{\partial y}\frac{\partial n}{\partial z} \\ \frac{\partial r}{\partial z}\frac{\partial n}{\partial x} & \frac{\partial r}{\partial z}\frac{\partial n}{\partial y} & \frac{\partial r}{\partial z}\frac{\partial n}{\partial z} \end{bmatrix}
\end{aligned} \tag{6.24}$$

where A, B, C are defined in Eq. 6.7.

Assuming that displacements vary linearly with time, we put this formula into the time integral $[\tau_{m-1}, \tau_m]$, and split the resulting integral into 3 parts:

$$\begin{aligned}
P^{nm} &= \int_{\tau_{m-1}}^{\tau_m} p^*_{lk} \cdot \left[\frac{\tau - \tau_{m-1}}{\tau_m - \tau_{m-1}} u_m + \frac{\tau_m - \tau}{\tau_m - \tau_{m-1}} u_{m-1} \right] \cdot d\tau \\
&= I_1 + I_2 + I_3
\end{aligned} \tag{6.25}$$

The first term is written as follows:

$$
\begin{aligned}
I_1 &= \int_{\tau_{m-1}}^{\tau_m^+} \frac{1}{4\pi} \cdot A \cdot \left\{ \left[\frac{\partial r}{\partial n} \cdot \delta_{ij} + r_{,l} r_{,k} \right] \right\} \left[\frac{\tau - \tau_{m-1}}{\tau_m - \tau_{m-1}} u_m + \frac{\tau_m - \tau}{\tau_m - \tau_{m-1}} u_{m-1} \right] d\tau \\
&= \left\{ \frac{6c_2^2}{r^4} \frac{1}{\tau_m - \tau_{m-1}} \left[a_1 \cdot u_m + a_2 \cdot u_{m-1} \right] \left(-\frac{3}{r^2} a_3 + \frac{c_2^2}{c_1^2} \frac{2}{r^2} a_4 + \frac{r}{c_2} a_7 \right) \cdot u_m \right. \\
&+ \left. \left(-\frac{3}{r^2} a_5 + \frac{c_2^2}{c_1^2} \frac{2}{r^2} a_6 - \frac{r}{c_2} a_7 \right) \cdot u_{m-1} \right\} \\
&\cdot \left[\frac{\partial r}{\partial n} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} \frac{\partial r}{\partial x}\frac{\partial n}{\partial x} & \frac{\partial r}{\partial x}\frac{\partial n}{\partial y} & \frac{\partial r}{\partial x}\frac{\partial n}{\partial z} \\ \frac{\partial r}{\partial y}\frac{\partial n}{\partial x} & \frac{\partial r}{\partial y}\frac{\partial n}{\partial y} & \frac{\partial r}{\partial y}\frac{\partial n}{\partial z} \\ \frac{\partial r}{\partial z}\frac{\partial n}{\partial x} & \frac{\partial r}{\partial z}\frac{\partial n}{\partial y} & \frac{\partial r}{\partial z}\frac{\partial n}{\partial z} \end{bmatrix} \right]
\end{aligned}
\tag{6.26}
$$

where

$$
\begin{aligned}
a_1 &= \left(-\frac{\tau^3}{3} + \frac{t + \tau_{m-1}}{2} \tau^2 - t \cdot \tau_{m-1} \tau \right) \Big|_{t_a}^{t_b} \\
a_2 &= \left(t_m \tau - \frac{t + \tau_m}{2} \tau^2 + \frac{\tau^3}{3} \right) \Big|_{t_a}^{t_b} \\
a_3 &= \frac{t - \frac{r}{c_2} - \tau_{m-1}}{\tau_m - \tau_{m-1}} \quad \text{and} \quad a_4 = \frac{t - \frac{r}{c_1} - \tau_{m-1}}{\tau_m - \tau_{m-1}} \\
a_5 &= \frac{\tau_m - \left(t - \frac{r}{c_2} \right)}{\tau_m - \tau_{m-1}} \quad \text{and} \quad a_6 \frac{\tau_m - \left(t - \frac{r}{c_1} \right)}{\tau_m - \tau_{m-1}} \\
a_7 &= \frac{1}{\tau_m - \tau_{m-1}}
\end{aligned}
\tag{6.27}
$$

where t is the current time, τ_{m-1} and τ_m are the times at step $m-1$ and m, r is the distance between the source point and the field point, t_a& t_b are the lower and upper limits of the time integral, a_i $(i = 1, ..., 7)$ are coefficients in the formula.

The second term is:

$$
\begin{aligned}
I_2 &= \int_{\tau_{m-1}}^{\tau_m^+} \frac{1}{4\pi}\cdot B\cdot\frac{\partial r}{\partial n}\cdot\begin{bmatrix} \frac{\partial r}{\partial x}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial x}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial x}\frac{\partial r}{\partial z} \\ \frac{\partial r}{\partial y}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial y}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial y}\frac{\partial r}{\partial z} \\ \frac{\partial r}{\partial z}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial z}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial z}\frac{\partial r}{\partial z} \end{bmatrix}\left[\frac{\tau-\tau_{m-1}}{\tau_m-\tau_{m-1}}u_m+\frac{\tau_m-\tau}{\tau_m-\tau_{m-1}}u_{m-1}\right]d\tau \\
&= \left\{-\frac{30c_2^2}{r^4}\frac{1}{\tau_m-\tau_{m-1}}\left[a_1\cdot u_m+a_2\cdot u_{m-1}\right]+\left(\frac{12}{r^2}a_3-\frac{c_2^2}{c_1^2}\frac{12}{r^2}a_4+\frac{2}{rc_2}a_7+\frac{2c_2^2}{rc_1^3}a_7\right)\cdot u_m\right. \\
&+ \left.\left(\frac{12}{r^2}a_5-\frac{c_2^2}{c_1^2}\frac{12}{r^2}a_6-\frac{2}{rc_2}a_7-\frac{2c_2^2}{rc_1^3}a_7\right)\cdot u_{m-1}\right\} \\
&\cdot \frac{1}{4\pi}\cdot\frac{\partial r}{\partial n}\cdot\begin{bmatrix} \frac{\partial r}{\partial x}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial x}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial x}\frac{\partial r}{\partial z} \\ \frac{\partial r}{\partial y}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial y}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial y}\frac{\partial r}{\partial z} \\ \frac{\partial r}{\partial z}\frac{\partial r}{\partial x} & \frac{\partial r}{\partial z}\frac{\partial r}{\partial y} & \frac{\partial r}{\partial z}\frac{\partial r}{\partial z} \end{bmatrix}
\end{aligned}
\tag{6.28}
$$

The third term is :

$$
\begin{aligned}
I_3 &= \int_{\tau_{m-1}}^{\tau_m^+} \frac{1}{4\pi}\cdot D\cdot\begin{bmatrix} \frac{\partial r}{\partial x}\frac{\partial n}{\partial x} & \frac{\partial r}{\partial x}\frac{\partial n}{\partial y} & \frac{\partial r}{\partial x}\frac{\partial n}{\partial z} \\ \frac{\partial r}{\partial y}\frac{\partial n}{\partial x} & \frac{\partial r}{\partial y}\frac{\partial n}{\partial y} & \frac{\partial r}{\partial y}\frac{\partial n}{\partial z} \\ \frac{\partial r}{\partial z}\frac{\partial n}{\partial x} & \frac{\partial r}{\partial z}\frac{\partial n}{\partial y} & \frac{\partial r}{\partial z}\frac{\partial n}{\partial z} \end{bmatrix}\left[\frac{\tau-\tau_{m-1}}{\tau_m-\tau_{m-1}}u_m+\frac{\tau_m-\tau}{\tau_m-\tau_{m-1}}u_{m-1}\right]d\tau \\
&= \frac{1}{4\pi}\begin{bmatrix} \frac{\partial r}{\partial x}\frac{\partial n}{\partial x} & \frac{\partial r}{\partial x}\frac{\partial n}{\partial y} & \frac{\partial r}{\partial x}\frac{\partial n}{\partial z} \\ \frac{\partial r}{\partial y}\frac{\partial n}{\partial x} & \frac{\partial r}{\partial y}\frac{\partial n}{\partial y} & \frac{\partial r}{\partial y}\frac{\partial n}{\partial z} \\ \frac{\partial r}{\partial z}\frac{\partial n}{\partial x} & \frac{\partial r}{\partial z}\frac{\partial n}{\partial y} & \frac{\partial r}{\partial z}\frac{\partial n}{\partial z} \end{bmatrix}\left\{\frac{6c_2^2}{r^4}\frac{1}{\tau_m-\tau_{m-1}}\left[a_1\cdot u_m+a_2\cdot u_{m-1}\right]\right. \\
&+ \left(-\frac{2}{r^2}a_3+\left(\frac{c_2^2}{c_1^2}\frac{2}{r^2}-\frac{1}{r^2}\left(1-2\frac{c_2^2}{c_1^2}\right)\right)\cdot a_4+\frac{1}{r^2}\left(1-2\frac{c_2^2}{c_1^2}\right)\frac{r}{c_1}a_7\right)\cdot u_m \\
&+ \left.\left(-\frac{2}{r^2}a_5+\left(\frac{c_2^2}{c_1^2}\frac{2}{r^2}-\frac{1}{r^2}\left(1-2\frac{c_2^2}{c_1^2}\right)\right)\cdot a_6+\frac{1}{r^2}\left(1-2\frac{c_2^2}{c_1^2}\right)\frac{r}{c_1}a_7\right)\cdot u_{m-1}\right\}
\end{aligned}
\tag{6.29}
$$

Eqs. 6.26, 6.28, 6.29 mean that we integrate the traction kernel p^* for point $\mathbf{x}^i$ at time t due to loading at point $\mathbf{x}^j$ between time τ_{m-1} and τ_m.

The physics of these integrals is illustrated by Fig. 6.2. At first, the element is quiescent since no signal from the field point has yet arrived. The time integral begins to have non-zero values when the pressure wave arrives at the element during the time interval $[\tau_{m-1}, \tau_m]$. The integral domain is the band between radii $c_1\tau_{m-1}$ and $c_1\cdot\tau_m$. The same thing happens to the shear wave. There are three cases to be

considered:

- Terms in the time integral associated with the pressure wave term $\delta\left(t-\frac{r}{c_1}\right)$

The integral domain is the shaded area between $[c_1\tau_{m-1},\ c_1\tau_m]$ in Fig. 6.2.

- Terms in the time integral associated with the shear wave term $\delta\left(t-\frac{r}{c_2}\right)$

The integral domain is the shaded area between $[c_2\tau_{m-1},\ c_2\tau_m]$ in Fig. 6.2.

- Terms in the time integral associated with both $H\left(t-\frac{r}{c_1}\right)+H\left(t-\frac{r}{c_2}\right)$

The integral domain is the area between $[c_2\tau_{m-1},\ c_1\tau_m]$ in Fig. 6.2.

After the shear wave passes, the element is quiescent and the time integral becomes zero again. For this reason, the system matrices are highly sparse, which is quite similar to the 3D wave BEM case. This special structure follows from the convolution of the Dirac delta function and Heaviside function in the integral equations. The integration does not extend over the whole boundary of the space-time cylinder, but only over a band within the surface of two propagating wave spheres. Details of implementation of the resulting time integral algorithm can be found in Section 6.4.

6.2.3 Space integration

We can distinguish between two kinds of space integration: non-singular and singular. In the former case, the distance r between the field point x_k and source point x_l on the surface element is not zero. In the latter case, the field point x_k may coincide with a source point x_l on the surface element.

6.2.3.1 Non-singular case

We consider first the non-singular case. From Eq. 6.11 and the time integrals Eq. 6.20 and Eq. 6.25, and after introducing the shape functions in space, the integrals involving the displacement Green's function u^*_{lk} and its derivative p^*_{lk} become:

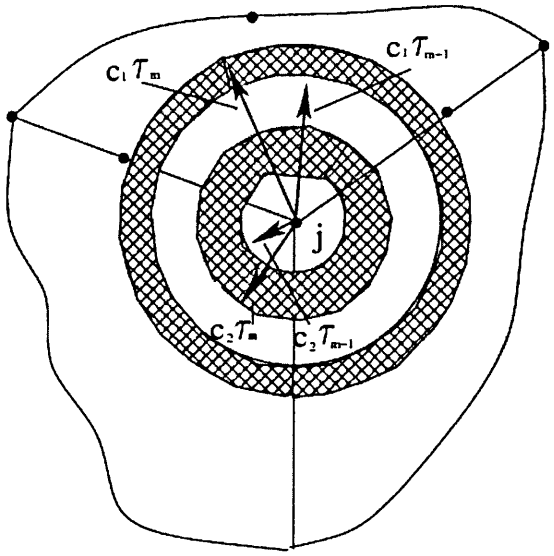


Figure 6.2: The band between radii $[c_1\tau_{m-1},\ c_1\tau_m]$ and $[c_2\tau_{m-1},\ c_2\tau_m]$ for the time integral

$$\begin{aligned} G_{lk}^{ij} &= \int_{S_j} u_{lk}^* \cdot N^j(\xi,\eta,\tau)\, dS_j(\boldsymbol{x},\tau) \\ &= \int_0^1 \int_0^1 U^{nm}(\mathbf{x}(\xi,\eta)) \cdot N^j(\xi,\eta)\, J(\xi,\eta)\, d\xi d\eta \\ &= \sum_{k=1}^{K} U^{nm}(\mathbf{x}(\xi_k,\eta_k))\, N^j(\xi_k,\eta_k)\, J(\xi_k,\eta_k)\, w_k \end{aligned} \tag{6.30}$$

$$\begin{aligned} \hat{H}_{lk}^{ij} &= \int_{S_j} p_{lk}^* \cdot N^j(\xi,\eta,\tau)\, dS_j(\boldsymbol{x},\tau) \\ &= \int_0^1 \int_0^1 P^{nm}(\mathbf{x}(\xi,\eta)) \cdot N^j(\xi,\eta)\, J(\xi,\eta)\, d\xi d\eta \\ &= \sum_{k=1}^{K} P^{nm}(\mathbf{x}(\xi_k,\eta_k))\, N^j(\xi_k,\eta_k)\, J(\xi_k,\eta_k)\, w_k \end{aligned} \tag{6.31}$$

where $U^{nm}(\mathbf{x}(\xi,\eta))$, $P^{nm}(\mathbf{x}(\xi,\eta))$ are time integrals of the displacement and traction kernel functions, $N^j(\xi_k,\eta_k)$ are shape functions for 6-node curved triangular elements, $J(\xi_k,\eta_k)$ is the Jacobian for the local coordinates, and w_k are the weight coefficients of Gauss quadrature, K is the total number of the Gauss points.

Because only some of elements are influenced at each time step, G_{lk}^{ij} and $\hat{H}_{lk}^{ij}$ in other elements are all zero. It is a waste of time to use brute force to loop over all elements to check whether they are zero or not. One approach is to check if the signal has reached the center of an element. If so, the whole element is considered to be affected (M. Marrero and J. Dominguez [55]). However, this approach introduce significant errors if elements are large. In order to improve accuracy, an element

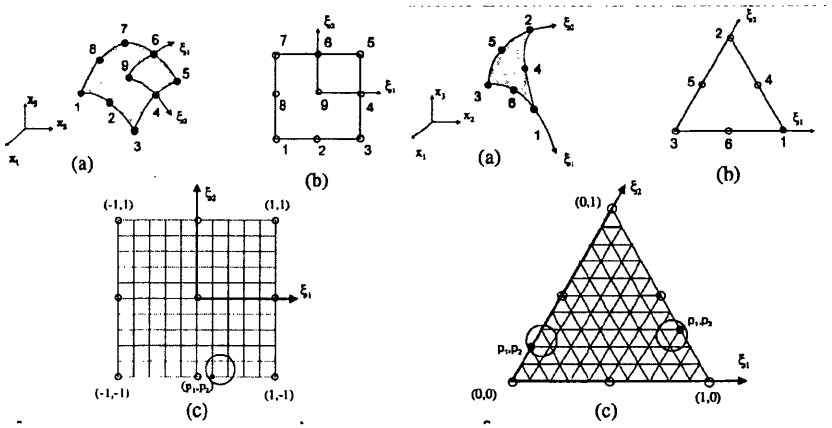


Figure 6.3: Subdivisions of a quadrilateral element and a triangular element (M. Marrero and J. Dominguez [55]) 2003)

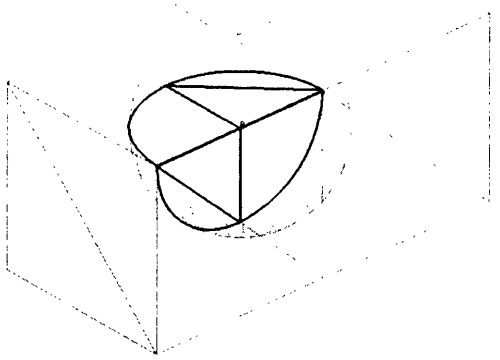


Figure 6.4: Intersection of triangular elements to decide integral areas exactly

may be divided into subelements as shown in Fig. 6.3. This method is simple, but can be very time-consuming.

Here, we introduce a more efficient method. When the wave sphere intersects with elements, the exact integral domain of the intersection is calculated and divided into several smaller curvilinear triangles in local coordinates as shown in Fig. 6.4. If the source node is not within the current triangular element, standard 9-point Gauss quadrature for the triangle is applied for each curved subtriangle, and then summed to complete the whole space integration. This process is applied to both pressure waves and shear waves. During the integration procedure, four spheres intersect the boundary surface, as shown in Fig. 6.2.

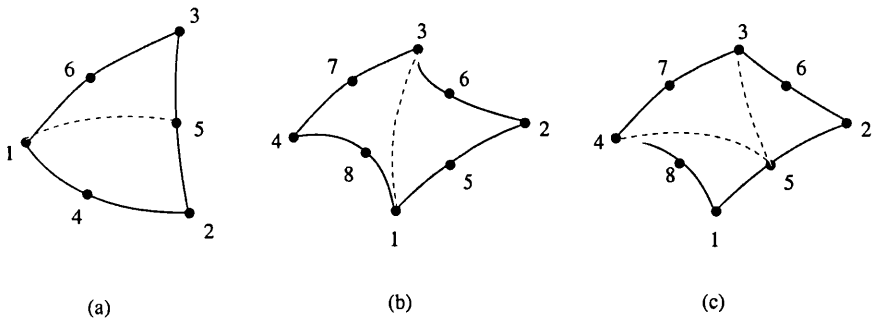


Figure 6.5: Element subdivision for singular integrals

6.2.3.2 Singular case

If we examine the displacement kernel $u^*_{lk}(\boldsymbol{x}, t; \boldsymbol{x}^i)$ in Eq. 6.5 and the traction kernel $p^*_{lk}(\boldsymbol{x}, t; \boldsymbol{x}^i)$ in Eq. 6.6, two kinds of singular integral need careful treatment. We note that there is a weak singularity of $O(1/r)$ in the displacement kernel, and a strong singularity of $O(1/r^2)$ in the traction kernel. Fortunately, the term in $1/r^3$ before the term $\left[H\left(t-\frac{r}{c_1}\right)-H\left(t-\frac{r}{c_2}\right)\right]$ is not singular because as $r \to 0$, $[H\left(t-\frac{r}{c_1}\right)-H\left(t-\frac{r}{c_2}\right)] \to 0$.

Singularity of the displacement kernel

The displacement singularity can be treated by employing an element subdivision technique to subdivide the original singular element into several triangular subelements. The common vertex of all subelements is the singular point, as shown in Fig. 6.5. In the case of a triangular element, either no subdivision is needed (vertex 1, 2, 3) or subdivision into 2 subelements (vertex 4, 5, 6). In the case of a rectangular element, we subdivide it into two subelements (vertex 1, 2, 3, 4) or 3 subelements (vertex 5, 6, 7, 8).

Then, each subtriangle can be mapped into square intrinsic element space as shown in in Fig. 6.6. Thus, nodes 1, 4 and 8 collapse into the same point P. As a result of this mapping, it can be shown that the Jacobian of the transformation is of order r, where r is the distance from the vertex P. Consequently, the weak singularity of $O(1/r)$ is nullified and the integral can be performed using normal

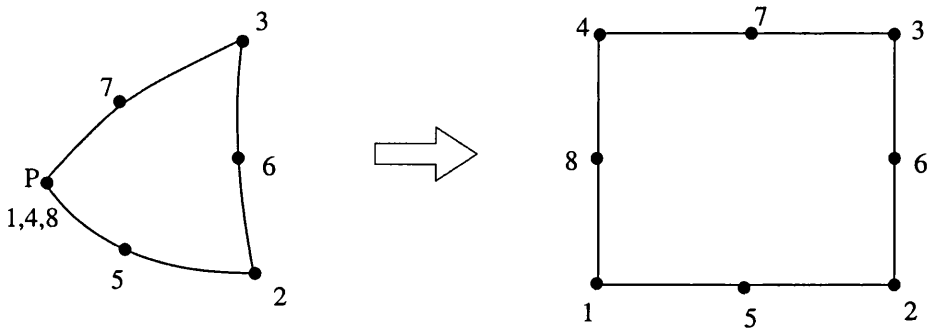


Figure 6.6: Transformation of a triangular subelement with singularity at node P to be a square

Gauss quadrature [27]. In general, the numerical quadrature can be written as:

$$\begin{aligned}\int_S f(\mathbf{x})dS &= \int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta)J(\xi,\eta)d\xi d\eta \\ &= \sum_{i=1}^{M} \int_{-1}^{1}\int_{-1}^{1} \sum_{k=1}^{8} N^k(\xi,\eta)f^k(\xi,\eta)J(\xi,\eta)d\xi d\eta \end{aligned} \tag{6.32}$$

where (ξ,η) are the local coordinates; $N^k(\xi,\eta)$ are quadratic shape functions for a 8-node rectangle; $J(\xi,\eta)$ is the Jacobian;$f^k(\xi,\eta)$ is the function value at the sampling point (ξ,η); M is the number of triangular subelements.

Singularity of the Traction kernel

For the traction singularity, an indirect method is used which employs the rigid body motion condition to calculate the sum of the c^i_{lk} coefficient and $[H]^{ii}_{kl}$ using Eq.6.13. The key idea behind the indirect scheme is that by prescribing a rigid body motion to the computational domain $\{u_i\} = \{1\}$, the corresponding tractions are all zeroes because rigid-body displacements do not cause any deformation or stresses. Then, the diagonal singular entries can be written in terms of the off-diagonal entries in the same row. In particular, if a rigid body motion is applied to Eq. 6.4, and all $p_k(\boldsymbol{x},\tau)$ are zero:

$$c^i_{lk}(x_i) + \sum_{q=1}^{Q} \int_{S_q} p^*_{lk}\left(\boldsymbol{x}, t-\tau; \boldsymbol{x}^i\right) dS(\boldsymbol{x},\tau) = 0 \tag{6.33}$$

where, $\boldsymbol{x_i}$ refers to a specific collocation point on S.

We can write it in the matrix form:

$$[H]_{kl}^{ii} = (\delta_{ij} - 1) \sum_{i=1}^{N} [H]_{kl}^{ij} \tag{6.34}$$

However, if we are dealing with infinite region problems, this strategy has to be modified. Since there are two boundaries to consider, (one is a finite part S, while the other is an infinite part S_∞), Eq. 6.33 becomes (Gao & Davies [27]):

$$c_{lk}^{i}(x_i) + \sum_{q=1}^{Q} \int_{S_q} p_{lk}^{*}\left(\boldsymbol{x}, t-\tau; \boldsymbol{x}^i\right) dS(\boldsymbol{x}, \tau) + \int_{S_\infty} p_{lk}^{*}\left(\boldsymbol{x}, t-\tau; \boldsymbol{x}^i\right) dS(\boldsymbol{x}, \tau) = 0 \tag{6.35}$$

The last integral on the left-hand side can be integrated analytically,

$$\int_{S_\infty} p_{lk}^{*}\left(\boldsymbol{x}, t-\tau; \boldsymbol{x}^i\right) dS(\boldsymbol{x}, \tau) = \delta_{lk} \tag{6.36}$$

Therefore, the diagonal submatrix of the traction kernel can be calculated from the equation:

$$[H]_{lk}^{ii} = \delta_{lk} + (\delta_{ij} - 1) \sum_{i=1}^{N} [H]_{kl}^{ij} \tag{6.37}$$

The same method can be applied to semi-infinite region problems, which leads to:

$$[H]_{lk}^{ii} = \frac{1}{2}\delta_{lk} + (\delta_{ij} - 1) \sum_{i=1}^{N} [H]_{kl}^{ij} \tag{6.38}$$

Solid angle

Another way to compute the singularity of the traction kernel is to compute the solid angle subtended by the local region around the collocation point. Since plane triangles are used to represent the geometry of the computational domain, the term $\partial r/\partial n = 0$ when the field point $\mathbf{x}_k$ and the source point $\mathbf{x}_l$ are on the same plane. The singular integral of the traction kernel is always zero except for the jump term

at the singular point $\mathbf{x}_l$. i.e.:

$$\sum_{q=1}^{Q} \int_{S_q} p_{lk}^{*}\left(\boldsymbol{x}, t-\tau; \boldsymbol{x}^{i}\right) dS(\boldsymbol{x}, \tau) = 0 \tag{6.39}$$

Therefore the diagonal singular entries are the jump terms $c_{lk}^{i}(x_i)$ only, which can be computed from the solid angle α from the following equation:

$$c_{lk}^{i}(x_i) = \frac{\alpha}{4\pi} \tag{6.40}$$

By looping over each node and element, all terms are calculated and assembled into matrices $[G]$ and $[H]$. Once the matrices $[G]$ and $[H]$ have been obtained, the resulting matrix equation is solved by using Gauss elimination or GMRES.

6.2.4 The calculation of boundary stress

Once the boundary element solution for the problem is obtained, the stresses σ_{lk} at any point on the boundary S can be determined by post-processing the displacement and traction nodal values in elements.

First, we introduce a local Cartesian coordinate system $(x_1^{'}, x_2^{'}, x_3^{'})$ in which $x_1^{'}$ and $x_2^{'}$ are tangential to the surface and $x_3^{'}$ is directed in the normal direction. Their relation to the general curved orthonormal basis (s_1, s_2, n) is shown in Fig. 6.7.

Denoting the displacement components in terms of the local coordinate system as $u_i^{'}$, the local tangential strains $\varepsilon_{11}^{'}, \varepsilon_{12}^{'} = \varepsilon_{21}^{'}$ and $\varepsilon_{22}^{'}$ components of the strain tensor at the point can be obtained as

$$\varepsilon_{\alpha\beta}^{'} = \frac{1}{2}\left(\frac{\partial u_{\alpha}^{'}}{\partial x_{\beta}^{'}} + \frac{\partial u_{\beta}^{'}}{\partial x_{\alpha}^{'}}\right) = \frac{1}{2}\left((s_{\alpha}\cdot e_m)\frac{\partial u_m}{\partial s_{\beta}} + (s_{\beta}\cdot e_m)\frac{\partial u_m}{\partial s_{\alpha}}\right) \tag{6.41}$$

Also, using Hooke's law and the definition of the traction vector, all the components of the symmetric stress tensor can be obtained (see Fig. 6.8):

$$\sigma_{33}^{'} = t_3^{'} = (n\cdot e_m)t_m$$

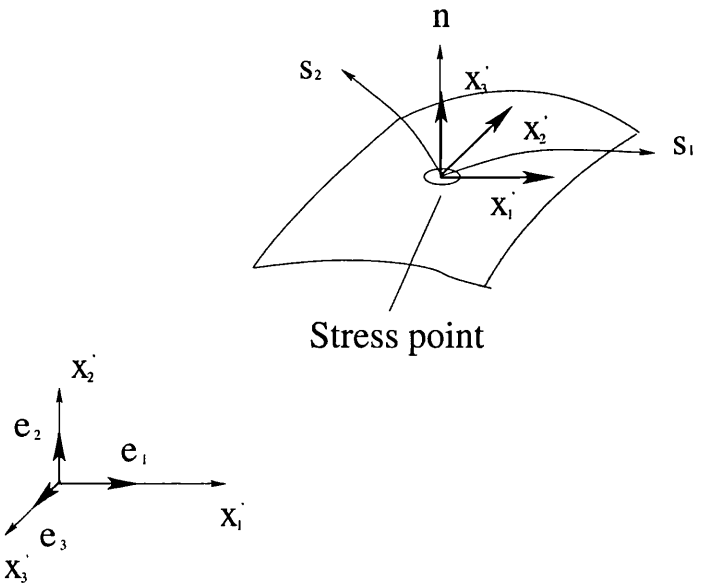


Figure 6.7: Local orthonormal coordinate system & global coordinates

$$\sigma'_{11} = \frac{1}{1-\nu}\left[2\mu\left(\varepsilon'_{11} + \nu\varepsilon'_{22}\right) + \nu\sigma'_{33}\right]$$

$$\sigma'_{22} = \frac{1}{1-\nu}\left[2\mu\left(\varepsilon'_{22} + \nu\varepsilon'_{11}\right) + \nu\sigma'_{33}\right]$$

$$\sigma'_{13} = \sigma'_{31} = t'_1 = (s_1 \cdot e_m)t_m$$

$$\sigma'_{23} = \sigma'_{32} = t'_2 = (s_2 \cdot e_m)t_m$$

$$\sigma'_{12} = \sigma'_{21} = 2\mu\varepsilon'_{12} \tag{6.42}$$

Thus, once the strain components $\varepsilon'_{\alpha\beta}$ and traction components t'_α are known, the computation of the stresses is straightforward. However, the computation of those components of the strain tensor is rather involved since the values of the tangential derivatives of the displacement $\partial u_i/\partial s_\alpha$ are not directly obtained from the BEM solution.

Using the results from K. H. Muci-Kuchler (2001) [56] , the tangential derivatives

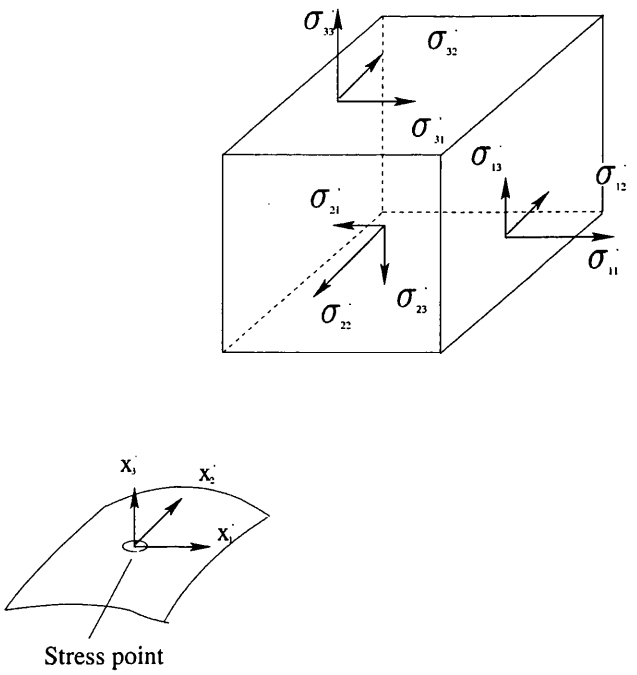


Figure 6.8: Components of the stress tensor in a local coordinate system

of the displacements are:

$$\frac{\partial u_i}{\partial s_\alpha} = \frac{1}{|J|}\left[\left(g_{22}\vec{d_1} - g_{12}\vec{d_2}\right)\frac{\partial u_i}{\partial \eta_1} + \left(g_{11}\vec{d_2} - g_{21}\vec{d_1}\right)\frac{\partial u_i}{\partial \eta_2}\right]\cdot s_\alpha \tag{6.43}$$

where $\vec{d_\alpha} = \left(\frac{\partial x_1}{\partial \eta_\alpha}, \frac{\partial x_2}{\partial \eta_\alpha}, \frac{\partial x_3}{\partial \eta_\alpha}\right)$, $g_{\alpha\beta} = \vec{d_\alpha}\cdot\vec{d_\beta}$, and $|J| = det\,(g_{\alpha\beta})$. In the boundary discretization, the values of $\frac{\partial u_i}{\partial \eta_\alpha}$ can be computed from the nodal values of the displacements and the derivatives of the shape functions with respect to the parametric coordinates η_1 and η_2

$$\frac{\partial u_i}{\partial \eta_\alpha} = \sum_{k=1}^{M}\frac{\partial N^k}{\partial \eta_\alpha}(\eta_1,\eta_2)\,u_l^k \tag{6.44}$$

where M is the number of the nodes in the element; N^k are the shape functions associated with the kth node, and u_l^k are values of the displacements at the kth node.

Once all the components of the stress tensors are determined in the local coordinate system, their counterparts in the global system can be obtained by using the

transformation law for second order tensors:

$$\sigma_{ij} = \left(e'_m \cdot e_i\right)\left(e'_n \cdot e_j\right)\sigma'_{mn} \tag{6.45}$$

6.3 Error estimates and adaptive schemes in 3D elastodynamics

As in the 3D wave case (see Chapter 5), the computation of error indicators in 3D elastodynamic problems is based on gradient-based error indicators, and based on the measuring the difference between two different numerical solutions of the boundary stresses.

6.3.1 Gradient-based error indicators

The gradient-based error indicator is based on the assumption that high gradients are strongly localized in space-time, and errors occur in the high-gradient areas. Thus, error indicators are derivatives of physical quantities within each element, which are normalized by their own pseudo-volumes in space-time. The difference between these error indicators and the corresponding 3D scalar wave error indicators is that the physical quantities here are vectors instead of scalars. The gradients of boundary stresses are used here rather than that of surface tractions.

The errors in an element are defined as:

$$e_u^{\alpha} = \left[\frac{\int\int\left[\left(\frac{\partial u_\alpha}{\partial x}\right)^2 + \left(\frac{\partial u_\alpha}{\partial y}\right)^2 + \left(\frac{\partial u_\alpha}{\partial t}\right)^2\right] dS_e dt}{\int\int dS_e dt}\right]^{\frac{1}{2}} \qquad (\alpha = 1,2,3)$$

$$e_\sigma^{\alpha\beta} = \left[\frac{\int\int\left[\left(\frac{\partial \sigma_{\alpha\beta}}{\partial x}\right)^2 + \left(\frac{\partial \sigma_{\alpha\beta}}{\partial y}\right)^2 + \left(\frac{\partial \sigma_{\alpha\beta}}{\partial t}\right)^2\right] dS_e dt}{\int\int dS_e dt}\right]^{\frac{1}{2}} \qquad (\alpha,\beta = 1,2,3) \tag{6.46}$$

where u_α, $\sigma_{\alpha\beta}$ are displacements and stresses obtained from the boundary mesh;

x, y are the local coordinates on the boundary; t is the time and S_e is the surface area of the boundary element. We can sum these errors, thus:

$$e_u = \sum_{\alpha=1}^{3} e_u^{\alpha} \qquad e_\sigma = \sum_{\alpha=1}^{3}\sum_{\beta=1}^{3} e_\sigma^{\alpha\beta} \tag{6.47}$$

6.3.2 Stress resolution based error indicators

We suppose that for a given problem, a sequence of approximate solutions for the boundary stresses may be calculated:

$$\hat{\sigma}_{ij}^{\{1\}}(\mathbf{x}), \hat{\sigma}_{ij}^{\{2\}}(\mathbf{x}), \hat{\sigma}_{ij}^{\{3\}}(\mathbf{x})...\hat{\sigma}_{ij}^{\{n\}}(\mathbf{x}) \tag{6.48}$$

where the solution $\hat{\sigma}_{ij}^{\{n\}}(\mathbf{x})$ has more degrees of freedom than $\hat{\sigma}_{ij}^{\{n-1\}}(\mathbf{x})$. As the number of degrees of freedom increases, then according to the saturation assumption [35], the solution with more degrees of freedom will be a better approximation of the exact solution. That is, the sequence converges to the true solution as the number of degrees of freedom increases:

$$\left|\sigma_{ij}(\mathbf{x}) - \hat{\sigma}_{ij}^{\{n\}}(\mathbf{x})\right| \to 0 \; as \; n \to \infty \tag{6.49}$$

where $\sigma_{ij}(\mathbf{x})$ is the exact solution, $\hat{\sigma}_{ij}^{\{n\}}(\mathbf{x})$ is the approximate solution.

If this is true, the two different approximate solutions in the sequence must approach arbitrarily close to each other, that is

$$\left|\hat{\sigma}_{ij}^{\{m\}}(\mathbf{x}) - \hat{\sigma}_{ij}^{\{n\}}(\mathbf{x})\right| \to 0 \; as \; n, m \to \infty \tag{6.50}$$

since

$$\begin{aligned} &\left|\sigma_{ij}(\mathbf{x}) - \hat{\sigma}_{ij}^{\{n\}}(\mathbf{x})\right| + \left|\sigma_{ij}(\mathbf{x}) - \hat{\sigma}_{ij}^{\{m\}}(\mathbf{x})\right| \\ \geq &\left|\sigma_{ij}(\mathbf{x}) - \hat{\sigma}_{ij}^{\{n\}}(\mathbf{x}) - \left[\sigma_{ij}(\mathbf{x}) - \hat{\sigma}_{ij}^{\{m\}}(\mathbf{x})\right]\right| \\ &= \left|\hat{\sigma}_{ij}^{\{m\}}(\mathbf{x}) - \hat{\sigma}_{ij}^{\{n\}}(\mathbf{x})\right| \end{aligned} \tag{6.51}$$

The two approximate solutions with different number of degrees of freedom can be chosen arbitrarily as long as they both fulfill the convergence requirement. Thus, an error estimate can be obtained from the difference between boundary stresses obtained from a coarse mesh and a refined one. The definition of error indicator for boundary stresses is:

$$e_\sigma = \left(\frac{\int\int \left(\triangle \sigma^{(e)}\right)^2 dS_e dt}{\int\int dS_e dt} \right)^{\frac{1}{2}} \tag{6.52}$$

Similarly, in discretized form,

$$e_\sigma = \left(\frac{\int\int \left(\triangle \sigma^{(e)}\right)^2 |J(\xi,\eta)| \, d\xi d\eta d\tau}{\int\int |J(\xi,\eta)| \, d\xi d\eta d\tau} \right)^{\frac{1}{2}} \tag{6.53}$$

Although theoretically any stress component can be used in the formulation above, quantities such as the Von Mises stress, the maximum principal stress, the minimum principal stress, or the maximum shear stress are good choices since they are more commonly used in the design of engineering structures to evaluate the risk of structural failure.

6.3.3 General adaptive algorithm

The general adaptive algorithm for 3D elastodynamic problems is the same as that described earlier for 3D scalar wave problems in Chapter 5. It follows the same process as before:

$$SOLVE \Rightarrow ESTIMATE \Rightarrow MARK \Rightarrow REFINE$$

Given the initial triangulation of the domain boundaries, we compute the solution vectors of displacements $\mathbf{u}_l$ and tractions $\mathbf{p}_l$ in the first step $SOLVE$. The solution error is estimated by postprocessing the displacements $\mathbf{u}_l$ and boundary stresses σ_{ij} results from the last step (this is the step of $ESTIMATE$). On the basis of the refinement indicators derived from the error estimates, the step $MARK$ identifies the elements in the current mesh in need of refinement. The new mesh is generated in the last step $REFINE$ and sent to the $SOLVE$ step in the next loop. The

iteration proceeds until the BEM solution satisfies the prescribed error tolerance. The general adaptive process is similar to that described in Chapter 4.

6.3.4 Local time stepping in 3D elastodynamics

Local time stepping in 3D elastodynamics is very similar to that for 3D scalar wave problems. The difference is that there are two values $\{u(x_i), q(x_i)\}$ at each vertex in wave problems while there are 6 values $\{u_l(x_i), {}_l(x_i)\}$ $(l = 1, 2, 3)$ at each vertex on the boundary in elastodynamic problems. The basic idea is to partition elements into N classes, which will be dynamically updated at each time step based on the error estimates; elements in the different classes k will be time-advanced using different local time steps $\triangle t/2^k$. The algorithm is briefly described as follows:

We denote the algorithm $R^N(\tau)$ for advancing class N in time over the time interval $\tau > 0$. We define $R^N(\tau)$ in a recursive way. The algorithm $R^0(\tau)$ describes how the original mesh advances one global time step $\triangle t$. For any $N \geq 0$, $R^{N+1}(\tau)$ is defined as follows:

1. Start with all boundary values of displacements $\mathbf{u}_l$ and tractions $\mathbf{p}_l$ known at time $t^n = n \triangle t$;

2. Advance all elements in class $k \leq N$ with $R^N(\triangle t)$ to obtain all unknown values of displacements $\mathbf{u}_l$ and tractions $\mathbf{p}_l$ at time $t^{n+1} = (n+1) \triangle t$: use the gradient-based error estimate to mark those elements with high-gradient values, which are labeled as class $N + 1$.

3. Refine the elements in class $N + 1$. Advance all elements in class $N + 1$ with $R^N(\triangle t/2)$. If required, use values at time t^{n+1} for elements in class N.

4. Since the elements in class $N + 1$ are solved twice so far, use the re-solution error estimate to mark those elements with large errors, which are labeled as class $N + 2$. For those elements with small errors, cancel its refinement in class $N + 1$: use the values in class N.

5. Refine the elements in class $N + 2$, Advance all elements in class $N + 2$ with $R^N(\triangle t/4)/$ If required, use values at time t^{n+1} for elements in class $k \geq N + 1$.

6. Continue this process of further refinement in both space and time. If we call the last refined class k, then advance the corresponding elements in class k with

$R^N(\triangle t/2^k)$. This process goes on until the prescribed error tolerance is reached.

7. After high-gradients of displacements $\mathbf{u}_l$ and tractions $\mathbf{p}_l$ are located and refinements are performed, for all element in class $k > 0$, finish the remaining time with $R^N(\triangle t/2^k)$ in one space-time BEM step to reach time step t^{n+1}.

8. At this stage, all unknown displacements $\mathbf{u}_l$ and tractions $\mathbf{p}_l$ at time t^{n+1} have been computed with local time stepping and satisfy the prescribed tolerance.

6.4 Numerical implementation

In this section, only aspects of the numerical implementation which are special for elastodynamics are explained. Because most of the programming of 3D elastodynamics is quite similar to that of 3D wave problems, what has been presented in Chapter 5 will not be repeated here.

6.4.1 Program structure

The adaptive boundary element program for 3D elastodynamics consists of five modules:

1. Geometry and boundary condition input module, mesh preprocessor

2. Compute the G_{ij}^e and H_{ij}^e— influence matrices of each element, assemble into general matrices $[G]$ and $[H]$.

3. Linear system solver

4. Compute the gradient-based and resolution-based error estimates, and mark those elements with large errors

5. Boundary element refinement module

The implementation for 3D elastodynamics requires operations on vector variables to compute the elements of influence matrices G_{ij}^e & H_{ij}^e; numerical integration of the weak singularity, and an adaptive algorithm with local time stepping. In the subsequent sections, we discuss some of the relevant programming aspects.

6.4.2 Numerical integral of the weak singularity

As mentioned in Section 6.2.3.2, the weak singularity of the displacement kernel (of $O(1/r)$) can be treated by employing an element subdivision technique to divide the original singular element into several triangular subelements. Then each subtriangle can be mapped into a rectangular element space for which the integral can be performed using normal Gauss quadrature. The algorithm is shown in Fig. 6.9. First, 6 nodes of a subtriangle are mapped into 8 nodes of a degenerated rectangle. However, the singular point must be one of the corner points of the rectangle. Then the normal 4×4 points Gauss quadrature is applied to compute the displacement kernel with the weak singularity.

6.4.3 Computing influence matrices G^e_{ij} and H^e_{ij}

Since all boundary values in the elastodynamics boundary integral equations are vector variables, the influence coefficients G^e_{ij} and H^e_{ij} are no longer scalar but are 3×3 matrices. If the number of degrees of freedom on the boundary is n, the total number of equations is $3n$. The algorithm is shown in Fig. 6.10.

Firstly, we loop over all vertexes on the surface mesh, and use two sphere pairs with radii ${r^1}_{min} = c_1 m \cdot dt$, $r^1_{max} = c_1 (m+1) \cdot dt)$ and ${r^2}_{min} = c_2 m \cdot dt$, $r^2_{max} = c_2 (m+1) \cdot dt)$ to intersect the boundary surface ($m = 0, ..., N-1$), where N is total number of time steps. The intersected sub-area of elements are put into the sub-mesh list of the vertex. Since the influence coefficients for one vertex are 3×3 matrices now, we loop over $K = 1, 3$ and $L = 1, 3$ to numerically integrate each entry of G_{ij} and H_{ij}, then assemble them into the general influence matrices $[G]$ and $[H]$ according to the relation in the Fig. 6.10. Because the true intersection area is computed as the difference between the bigger sphere and smaller one, the *add-submesh* means the intersection with the bigger sphere while the *subtract-submesh* means the intersection with the smaller one. After finishing the integration with the *add-submesh*, we do the same thing with the *subtract-submesh*, but subtract the values from the general influence matrices $[G]$ and $[H]$.

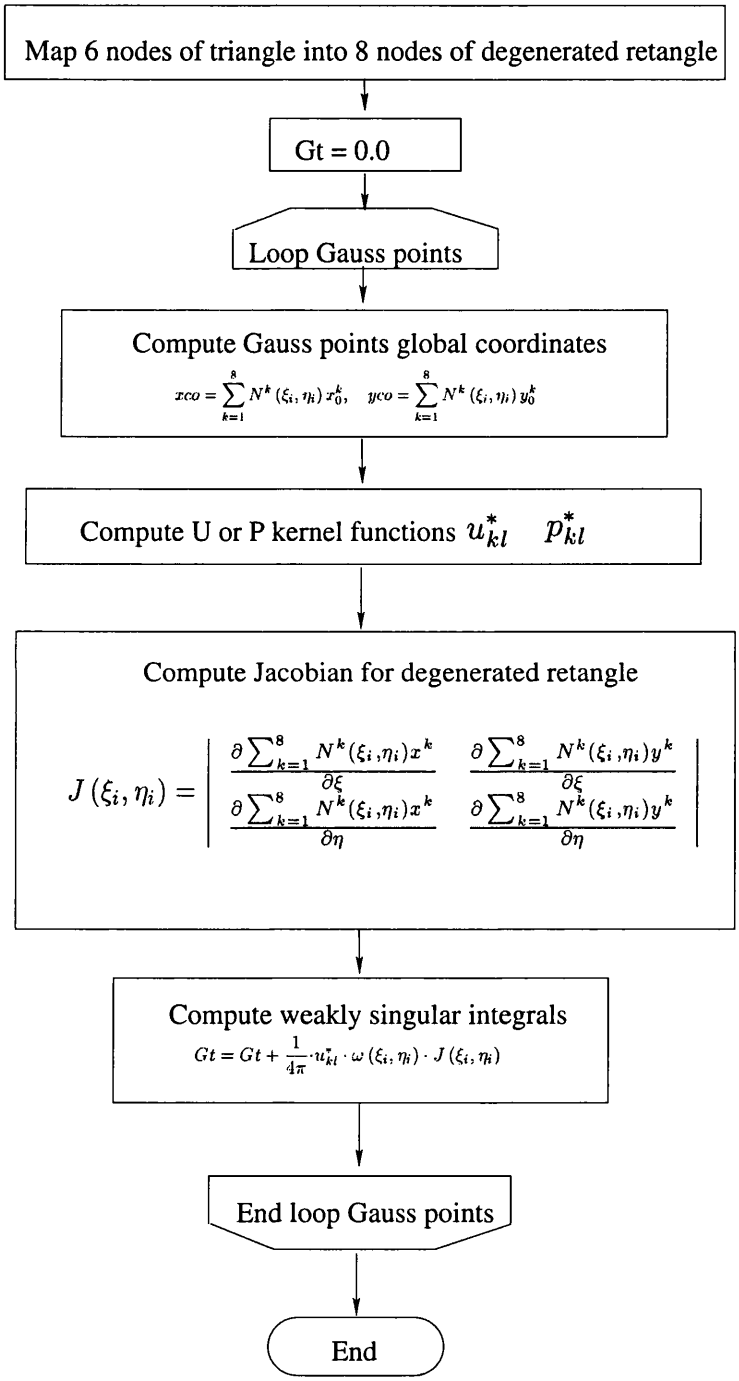


Figure 6.9: Numerical integration of the weak singularity

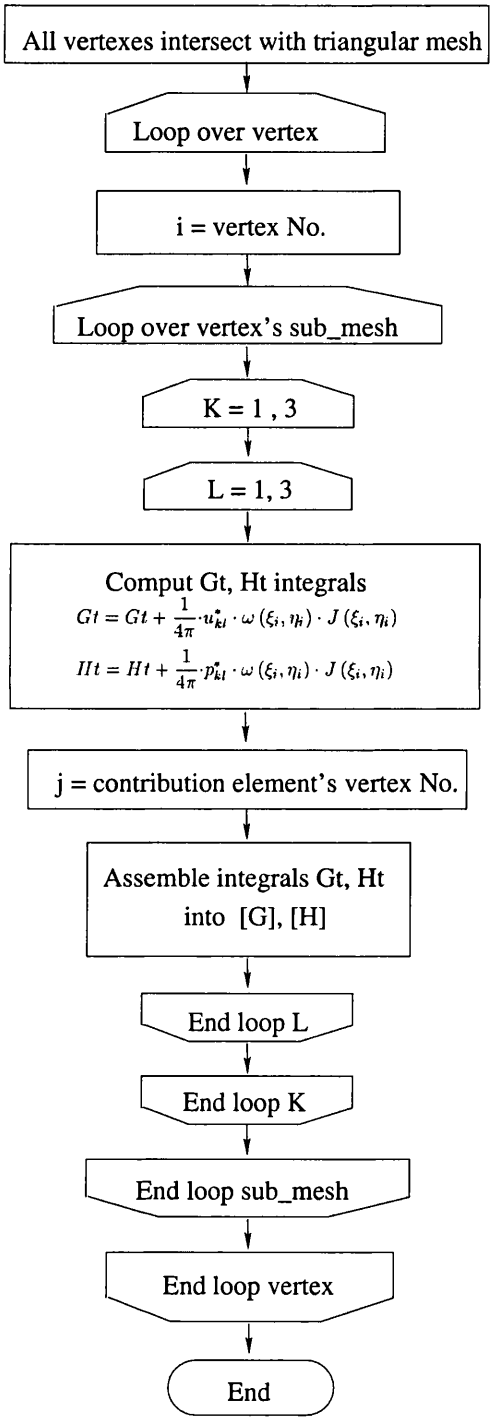


Figure 6.10: Computing elements of influence matrices G^e_{ij} and H^e_{ij}

6.4.4 Adaptive algorithm with local time stepping

As mentioned in Section 6.3.4, local time stepping involves partitioning elements into N classes which are dynamically updated at each time step based on error estimation: then elements in different classes k are time-advanced using different local time steps $\triangle t/2^k$. The algorithm is shown in Fig. 6.12.

The original uniform mesh is called the *root_ mesh* and labeled as class 0. The *root_ mesh* is advanced one step $\triangle t$ from t^n to t^{n+1} and the gradient-based error indicators are used to mark those element with large errors. We refine those marked elements and label them as class 1. The boundary integral equations are applied to those vertexes in elements of class 1 only with local time step set to $\triangle t/2$. If required, we use the boundary values at time t^{n+1} for elements in class 0. After solving the displacements $\mathbf{u}_l$ and tractions $\mathbf{p}_l$, we use the resolution error estimate to compute element errors and the global error. If the global error is bigger than the prescribed tolerance, elements with large errors are marked and refined as classes $k(k = 2, 3, ...)$ while the refinement is removed from elements with smaller errors. The boundary integral equations are applied to those vertexes in elements of class k only with local time steps: $\triangle t/2^k$. If required, we use boundary values at time t^{n+1} for elements in class $(0, 1, ..., k-1)$. The same process continues until the global error is smaller than the prescribed tolerance. The space-time BEM equation is applied for all elements in the refinement class $(1, 2, ..., k)$ with time step $(\triangle t/2^k)$ until t^{n+1} is reached. Thus, all elements are refined if necessary to satisfy the error tolerance while using their own local time stepping. The process is repeated until the final time is reached.

6.5 Numerical Examples

In this section, we present three numerical examples of 3D elastodynamic wave propagation problems.

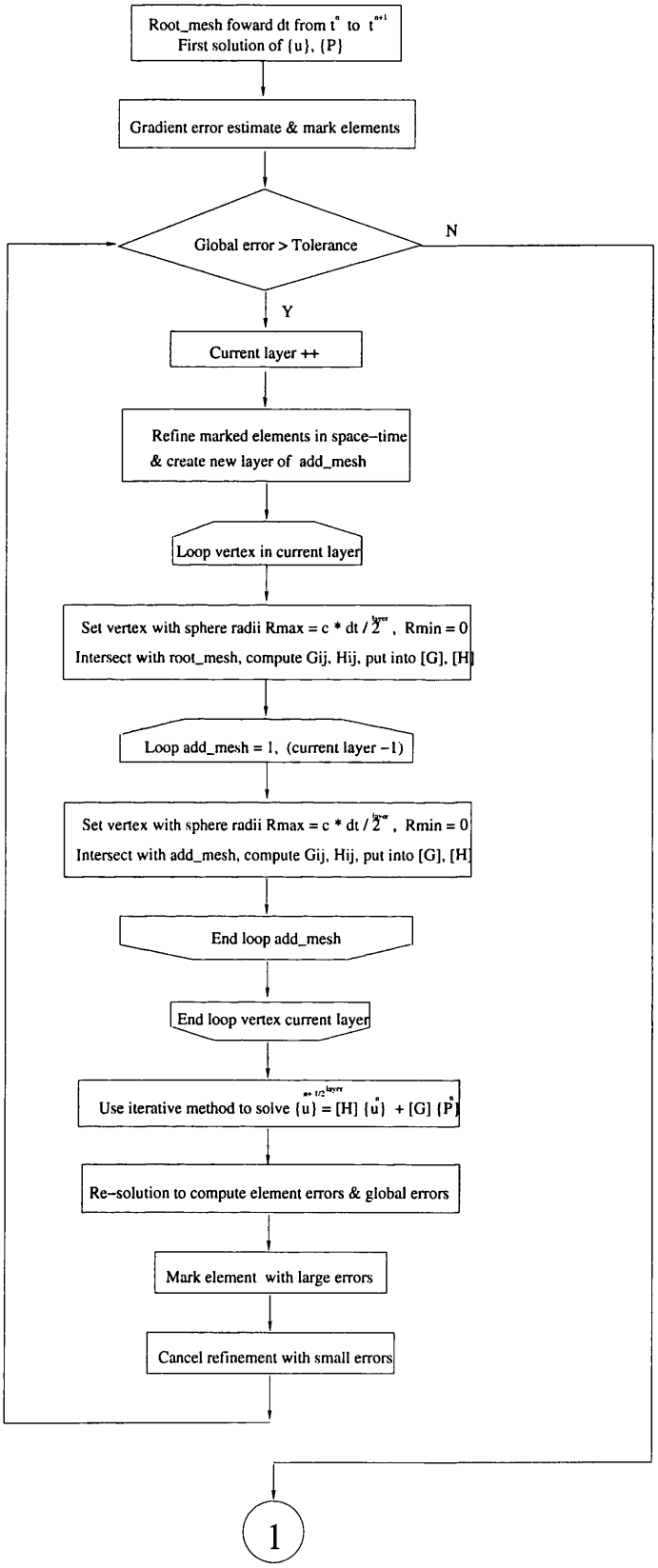


Figure 6.11: Adaptive algorithm with local time stepping, part 1

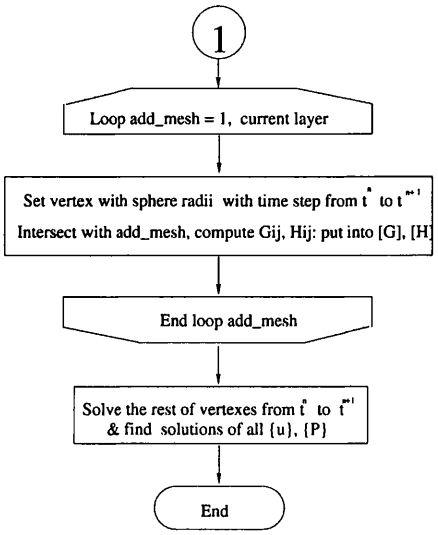


Figure 6.12: Adaptive algorithm with local time stepping, part 2

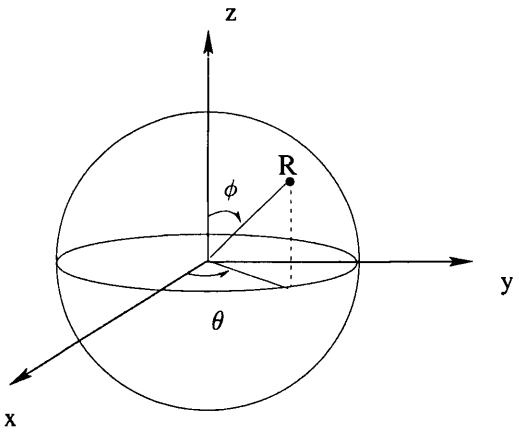


Figure 6.13: Spherical cavity embedded in an infinite continuum

6.5.1 Explosion inside a spherical cavity

The problem of a spherical cavity in infinite space subjected to explosive pressure is modeled by a suddenly applied pressure (Heaviside function). An analytical solution for the displacements and the stresses is available for comparison. Here we assume that the radius of spherical cavity $R = 5.38m$, the density of the surrounding material $\rho = 2.67 \times 10^3 kg/m^3$ and Poisson's ratio $\nu = 0.25$. The geometry and coordinate system is shown in Fig. 6.13.

The pressure wave speed $c_1 = 527.45m/s$, and the shear wave speed $c_2 = 303.95m/s$. An internal radial pressure $p_0 = 6.90 \times 10^6 Pa$ is suddenly applied and sustained thereafter. We compute the dynamic response of the spherical cavity

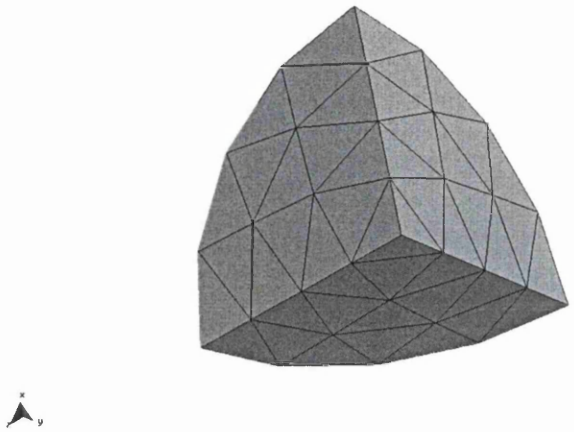


Figure 6.14: Mesh of an octant of the spherical cavity with 130 nodes, 64 triangular elements

wall under this loading, that is, the time history of the resulting displacements and stresses. Because of symmetry, we only model one octant of the spherical surface, using 64 curved triangular elements and 130 nodes, as shown in Fig. 6.14. The zero normal displacement boundary condition is applied on the shear-free planes of symmetry.

Fig. 6.15 shows the time history of the radial displacement of the cavity wall. The y axis is the radial displacement normalized by their static values; the x axis is the time normalized by the time taken for the pressure wave to travel the distance of the sphere radius R. The accuracy of the time-domain BEM solver is evident when the results are compared with the analytic solution (Timoshenko & Goodier [83]1970). For the displacement, the maximum error is around 2%. The time step $dt = 0.004$s; space-time ratio $\beta = 0.4$. In total there are 27 time steps.

If the time step is too large, for example, $\triangle t > 0.008$s, or $\beta > 0.8$, the solution fails to capture the peak value of the dynamic response, as shown in Fig. 6.16. If the time step is too small, $\triangle t < 0.002$s, or $\beta < 0.2$, the solution becomes unstable, as shown in Fig. 6.17. Therefore, the optimal space-time ratio β is between 0.3 and 0.7.

To study the dynamic response of the spherical cavity under different load histories, the normalized radial displacements histories at the cavity wall are computed for loads with different time variations, as shown in Fig. 6.18. These are: (a) a

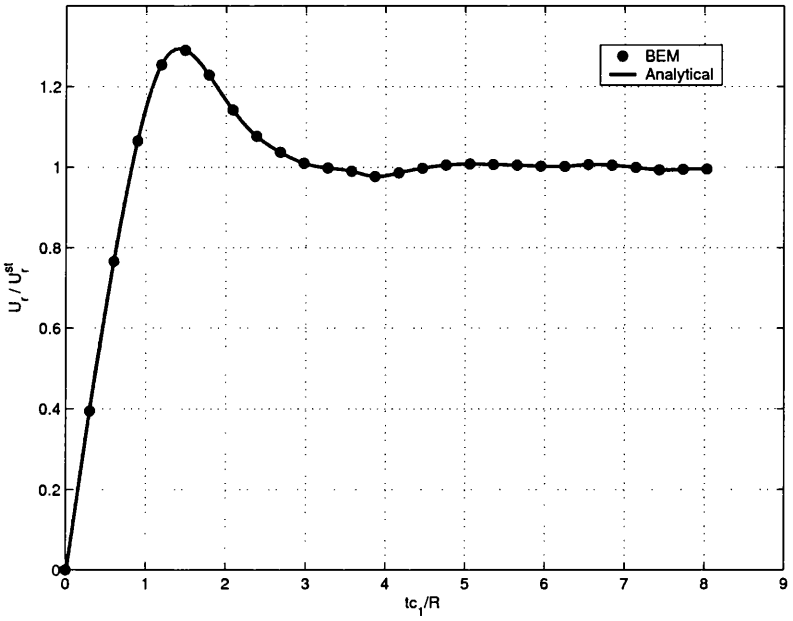


Figure 6.15: Normalized radial displacement time history of the spherical cavity

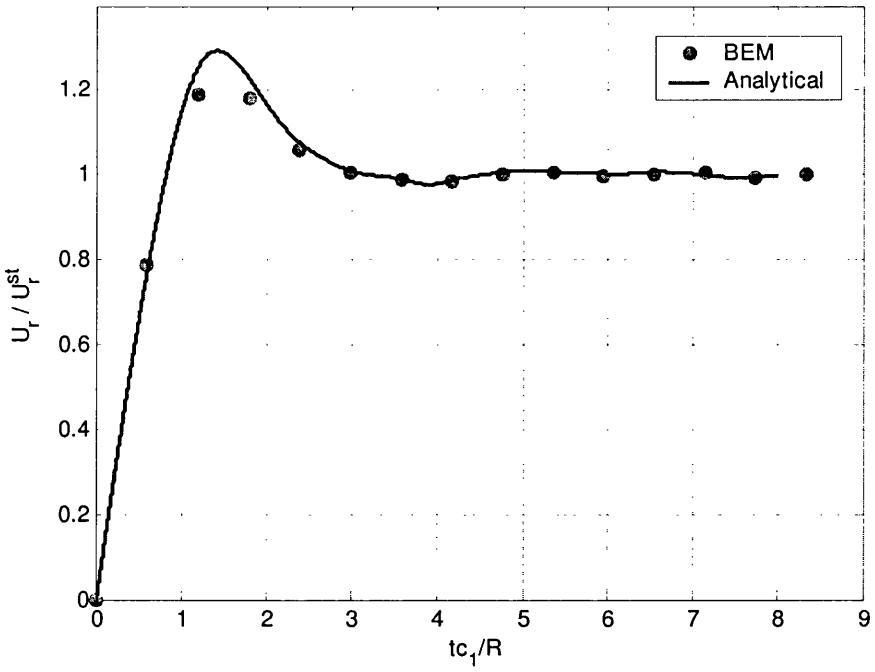


Figure 6.16: Normalized radial displacement time history of the spherical cavity, $\beta = 0.8$, $\triangle h = 2$ m, $\triangle t = 0.008$ s

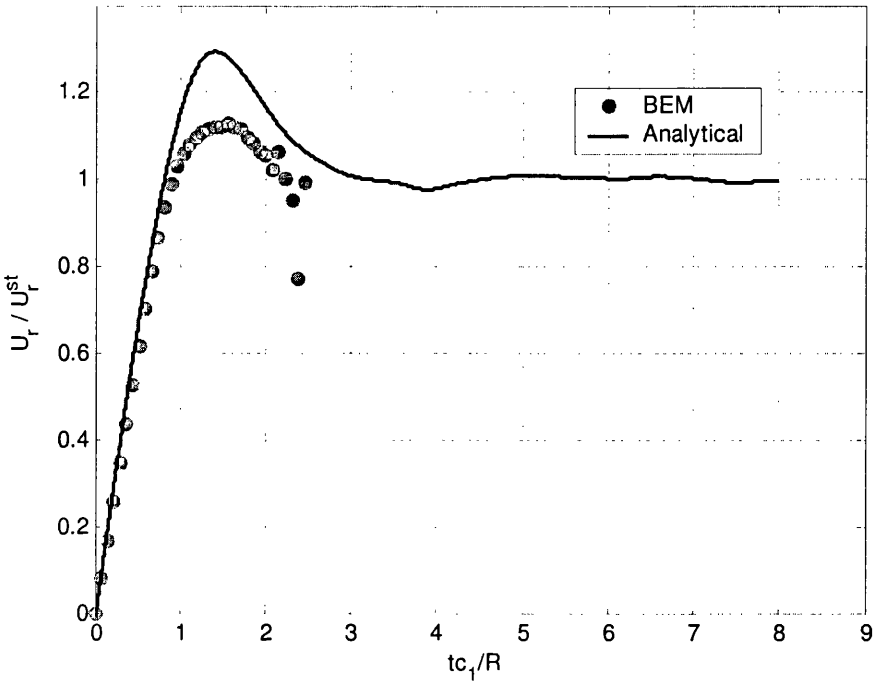


Figure 6.17: Normalized radial displacement time history of the spherical cavity, $\beta = 0.1$, $\triangle h = 2$ m, $\triangle t = 0.001$ s

ramp-type load with linear rise until time $t_0 = 2c_1 t/R$ is reached; (b) an exponential time rising load with the form $1 - e^{-t/t_0}$; (c) a rectangular pulse load until time t_0 is reached; (d) a triangular pulse load with linear rise and fall at the turning point $t_0/2$; (e) a Gaussian-type pulse load with the form $\frac{1}{\sqrt{2\pi}\sigma} exp\left[\frac{-(t-t_0/2)^2}{2\sigma^2}\right]$, $\sigma = t_0/6$.

The normalized radial displacement histories at the cavity wall for these different loads are plotted in Fig. 6.19. The reference response is the response under the Heaviside load. The response in the case of the ramp load resembles the reference one: it just reaches the peak value much later and then converges to the static solution. The response to the exponential increasing load just follows the load itself, gradually increasing to the static solution. The response of the rectangular load follows the first part of the Heaviside one, and then drops to zero. The response of the last two cases are similar to each other; both rise to their peak values and drop to zero. The difference is that rectangular one drops to zero slower than the triangular one since more energy is released in the former case. It is evidence of the accuracy and robustness of the algorithm that while the response from maintained-type loads

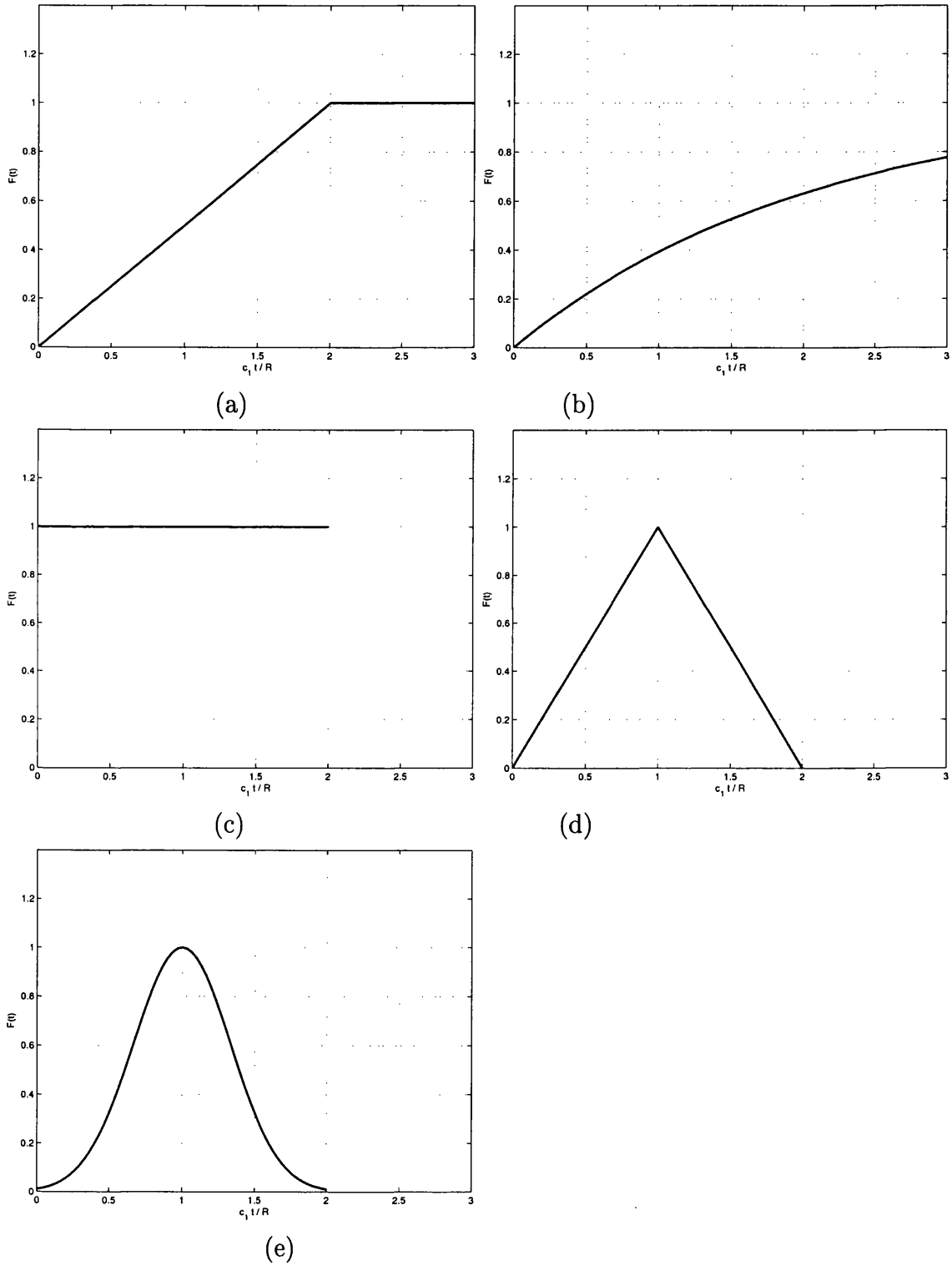


Figure 6.18: Loads with different time variation

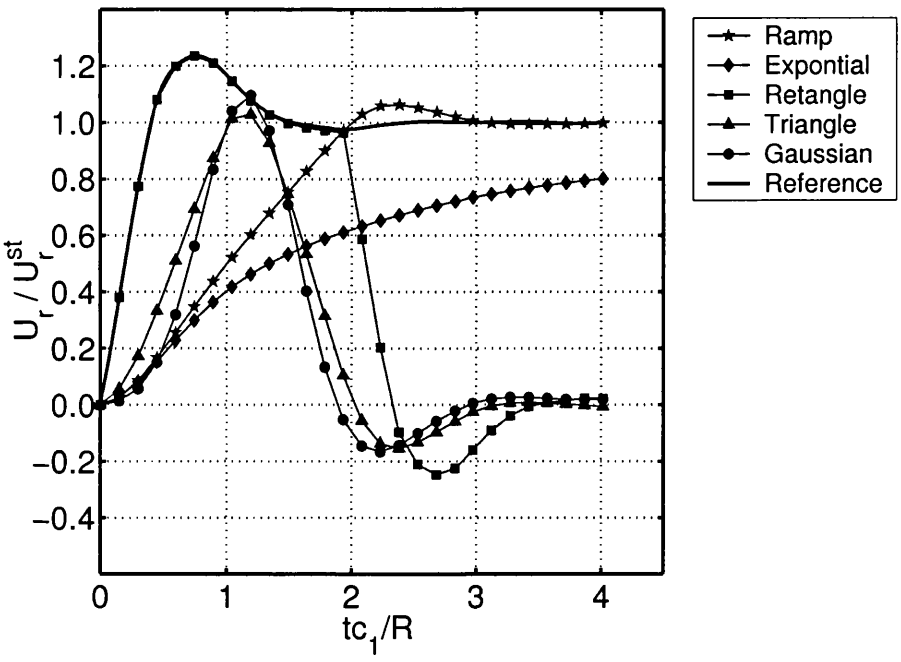


Figure 6.19: Normalized radial displacement history for various loads

converge to the static solution, while the response from impulse-type loads settles back to the quiescent state quickly after one or two oscillations.

It is interesting to note that in none of these cases does the peak radial displacement exceed the corresponding state value by a significant margin.

6.5.2 Rigid surface foundation under external load

We now explore the dynamic response of a rigid surface foundation supported by a linear elastic, homogeneous half-space and subjected to an external dynamic force. The dynamic interaction between soil and structures is of course very relevant to problems of geotechnical seismic design. The objectives of this numerical example are to find:

- The time history of the displacement of the rigid foundation under different loads, such as harmonic and impulsive loads;
- The traction distribution under the square foundation;
- The influence on stability and accuracy of the algorithm due to different element sizes, time-space ratios β , etc. ;

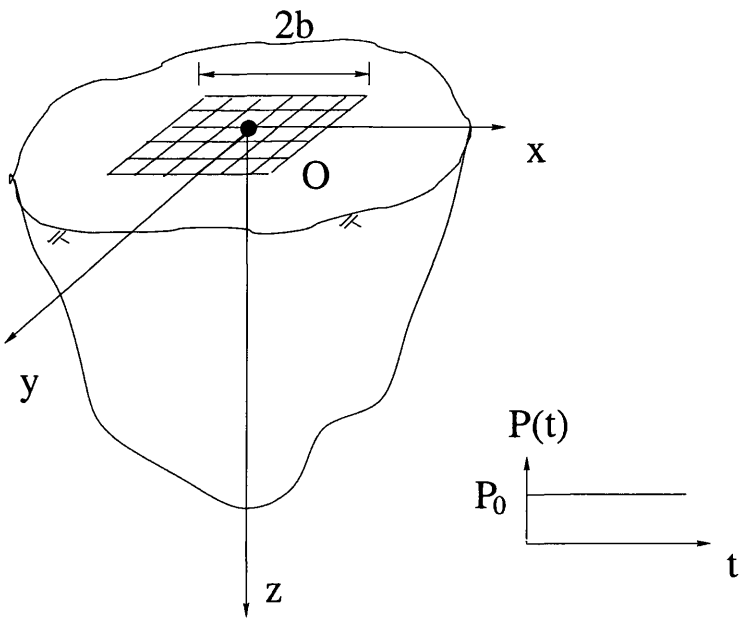


Figure 6.20: Geometry and coordinate system of a rigid surface foundation

- The compliance of the foundation under the different frequencies of harmonic loads.

The foundation is a square of dimension $2b = 15m$. The geometry and coordinate system is shown as Fig. 6.20. The foundation is assumed to be a rigid massless body.

The soil's material properties are $\lambda = 6.623 \times 10^6$, $\mu = 3.315 \times 10^6$, and $\rho = 2.82 \times 10^2 kg/m^3$, in consistent units. A vertical centric dynamic force is applied to the foundation. We assume that the foundation and the soil do not separate from each other during the motion. We compute the wave speeds:

$$c_1 = \sqrt{\frac{\lambda + 2\mu}{\rho}}$$

$$c_2 = \sqrt{\frac{\mu}{\rho}} \tag{6.54}$$

which yields a pressure wave speed $c_1 = 2.168 \times 10^2 m/s$, and a shear wave speed $c_2 = 1.084 \times 10^2 m/s$. The numerical test program involves:

(a) The soil-foundation interface is modeled by 36 triangular elements and 85 vertexes (Fig. 6.21). Without refinement, we compute the time history of the

displacement of the foundation due to harmonic and impulsive loads;

The traction distribution on the soil-foundation;

The time history of the displacement of the foundation for a series of time-space ratio $\beta = 0.1,\ 0.3,\ 0.5,\ 0.8,\ 1.2$ to find its influence on stability and accuracy;

The compliance of the rigid foundation;

(b) The mesh is now dynamically refined at each time step. We compute the time history of the displacement of the foundation due to harmonic and impulsive loads using refinement schemes. The results are then compared with the previous set of results.

(c) A larger foundation ($2b = 20m$) is modeled by 64 triangular elements and 145 vertexes (Fig. 6.21(b)). We compute the time history of the displacement of the foundation due to harmonic and impulsive loads.

Before proceeding to the numerical results, some discussion of the technique employed to implement the mixed boundary condition is needed. The problem is that neither displacements nor tractions are specified. However, the displacements are subject to the condition that they all are equal, while the tractions are subject to the condition that their integral is equal to the prescribed force $F(t)$. The BEM equation is:

$$H^{nn}u^n = G^{nn}p^n + \sum_{m=1}^{n-1} G^{nm}p^m - H^{nm}u^m \tag{6.55}$$

where n is the current time step, m is the past time step.

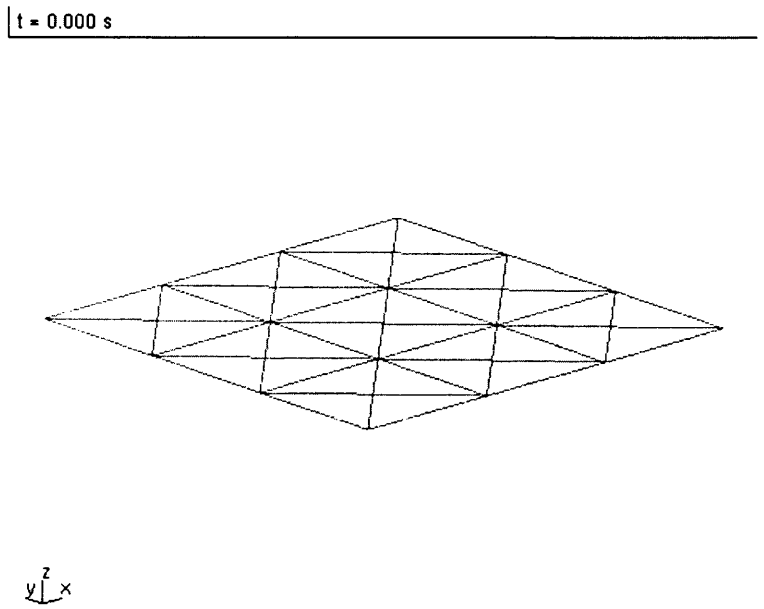
The integral of the traction on the interface is equal to the external force $F(t)$:

$$\int_{\Gamma} p^n(x,t)d\Gamma(x) = F(t) \tag{6.56}$$

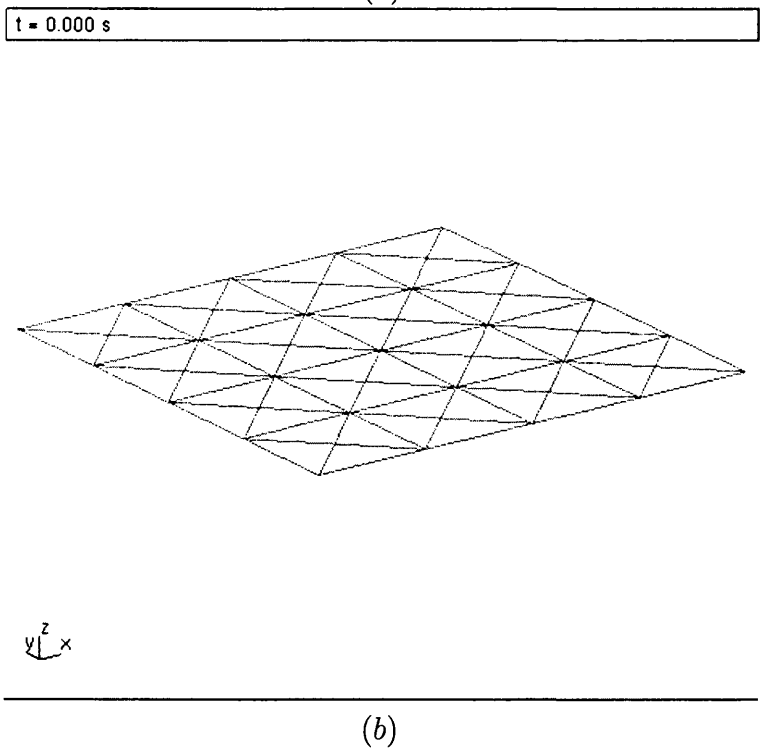
Its discrete form is:

$$[A] \cdot \{p^n\} = F_t \tag{6.57}$$

where $[A]$ is the discrete form of the integral operator and $F(t)$ is the external dynamic load applied on the foundation.



(a)



(b)

Figure 6.21: Discretization mesh of soil-rigid foundation (a) Original mesh 3x3 (b) a bigger foundation with mesh 4x4

We rewrite Eq. 6.55 and Eq. 6.57 in the matrix form:

$$\begin{bmatrix} H^{nn} & -G^{nn} \\ 0 & A \end{bmatrix}_{(N+1)\times 2N} \cdot \begin{Bmatrix} u^n \\ p^n \end{Bmatrix}_{2N\times 1} = \begin{Bmatrix} \sum_{m=1}^{n-1} G^{nm}p^m - H^{nm}u^m \\ F_t \end{Bmatrix}_{2N\times 1} \tag{6.58}$$

where N is the total number of degrees of freedom of the original BEM problem. Now the problem becomes a linear system with one more unknown.

Thus, we prescribe the external load $F(t)$, and solve Eq. 6.58 step by step through time to calculate the tractions $p(t)$ and displacements $u(t)$ until the final time is reached.

For case (a), we compute the time history of the displacement of the foundation due to harmonic and impulsive loads with different β, as shown in Fig. 6.22 and Fig. 6.23. The dynamic responses $u(t)$ exhibits a phase change relative to the input harmonic loads, normally represented by 'complex number' functions. As β values grow, the solutions become less and less accurate due to the numerical damping. However, when β is too small ($\beta = 0.1$), the solution becomes unstable, as shown in both the harmonic load and the impulse load case.

The traction distribution on the soil-foundation interface is shown in Fig. 6.24. Just as elastic theory predicts, singularities arise along the edges of the rigid foundation.

The refinement scheme is then applied to this numerical example. The refinement is invoked as the wave fronts move along the surface of the rigid foundation. The time history of the displacement of the foundation due to harmonic and impulsive loads for various values of β, as shown in Fig. 6.25 and Fig. 6.26. It is observed that the refinement scheme improves accuracy for almost all cases except when $\beta = 0.1$. In this case, because the time step is too small for refinement scheme to be stable, the refinement scheme actually accelerate the process of instability. With the adaptive scheme, we can start with a coarse mesh and obtain a much more accurate solution, as the case for $\beta = 0.8$ and $\beta = 1.2$.

To obtain the efficiency of adaptive BEM, we record the computing time for each simulation with the time-space ratio β from 0.1 to 1.2. (Table 6.1). For the adaptive

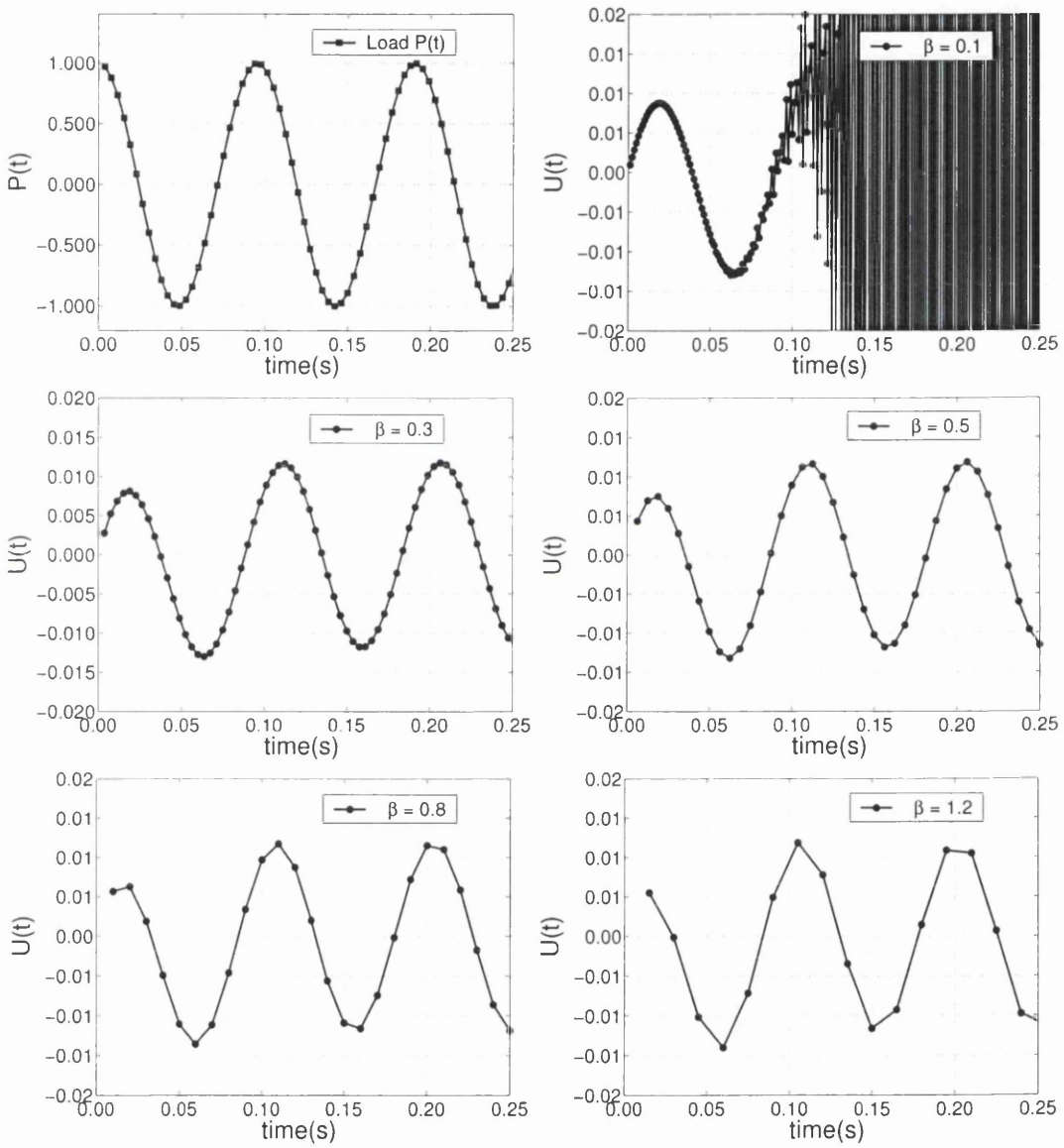


Figure 6.22: Time history of the displacement of the foundation due to harmonic loads with $\beta = 0.1,\ 0.3,\ 0.5,\ 0.8,\ 1.2$

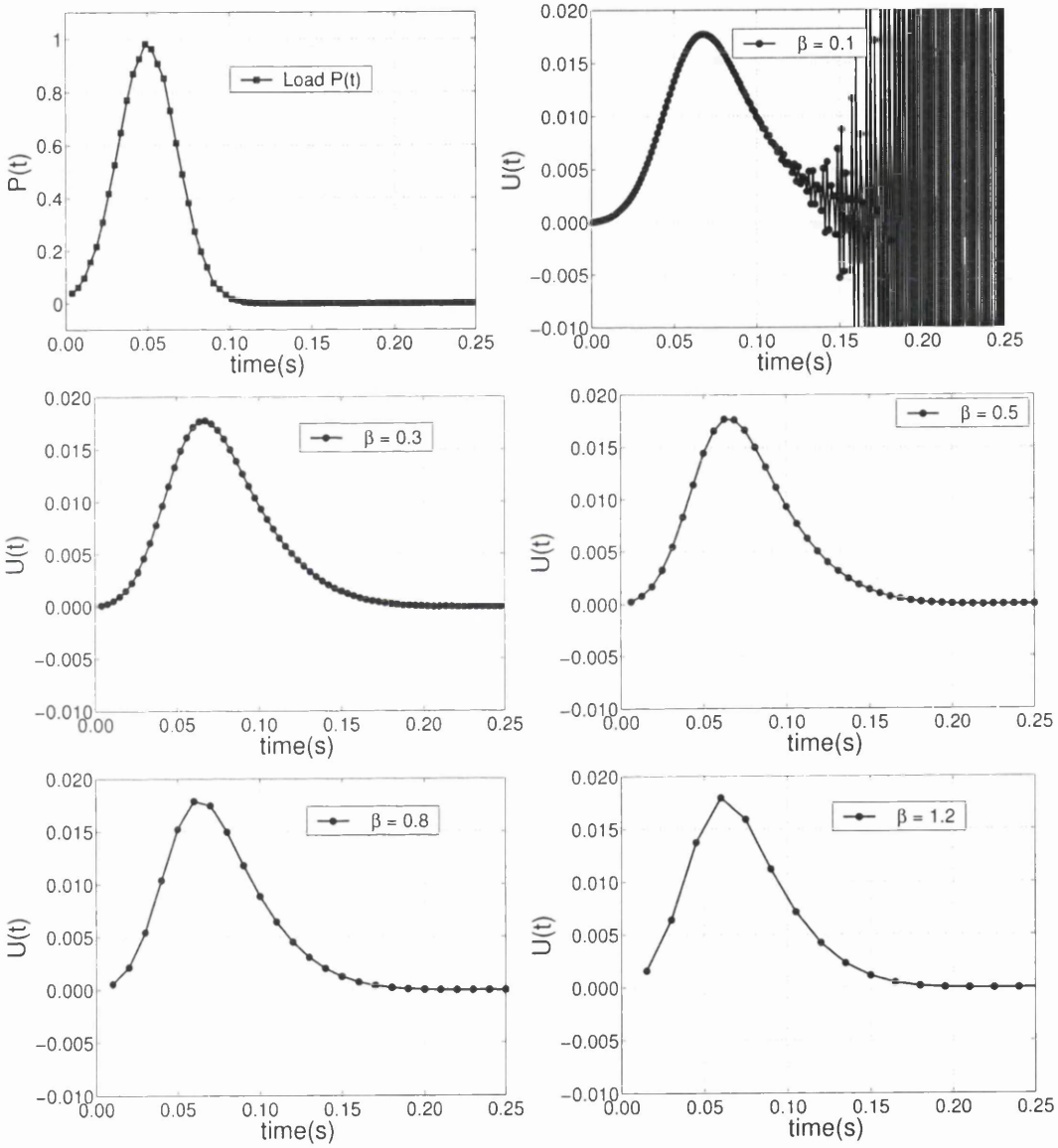


Figure 6.23: Time history of the displacement of the foundation due to impulse loads with $\beta = 0.1,\ 0.3,\ 0.5,\ 0.8,\ 1.2$

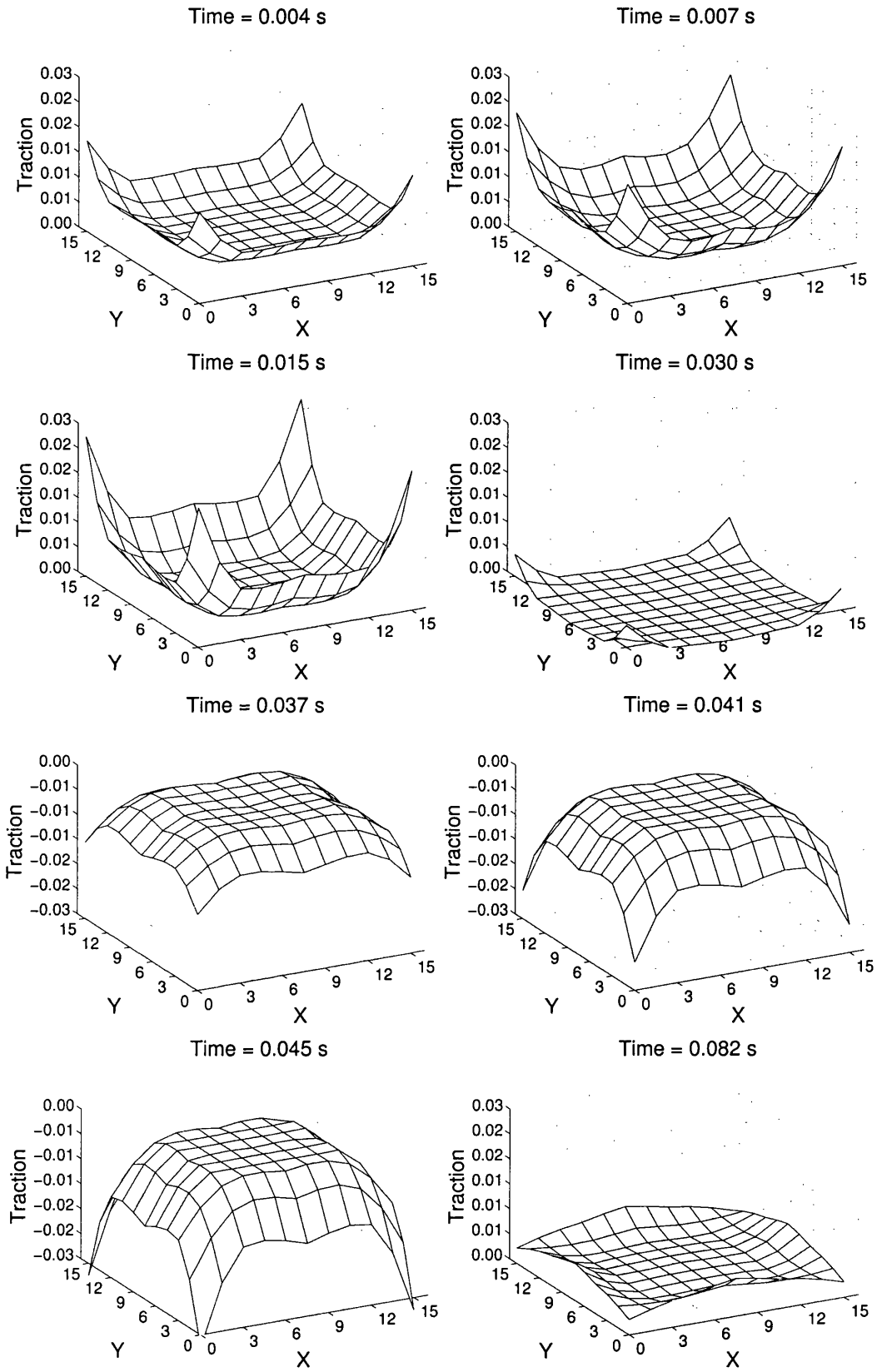


Figure 6.24: Traction distribution on the soil-foundation interface with harmonic load at time $t = 0.004$, 0.007, 0.015, 0.030, 0.0037, 0.041, 0.045, 0.082 s

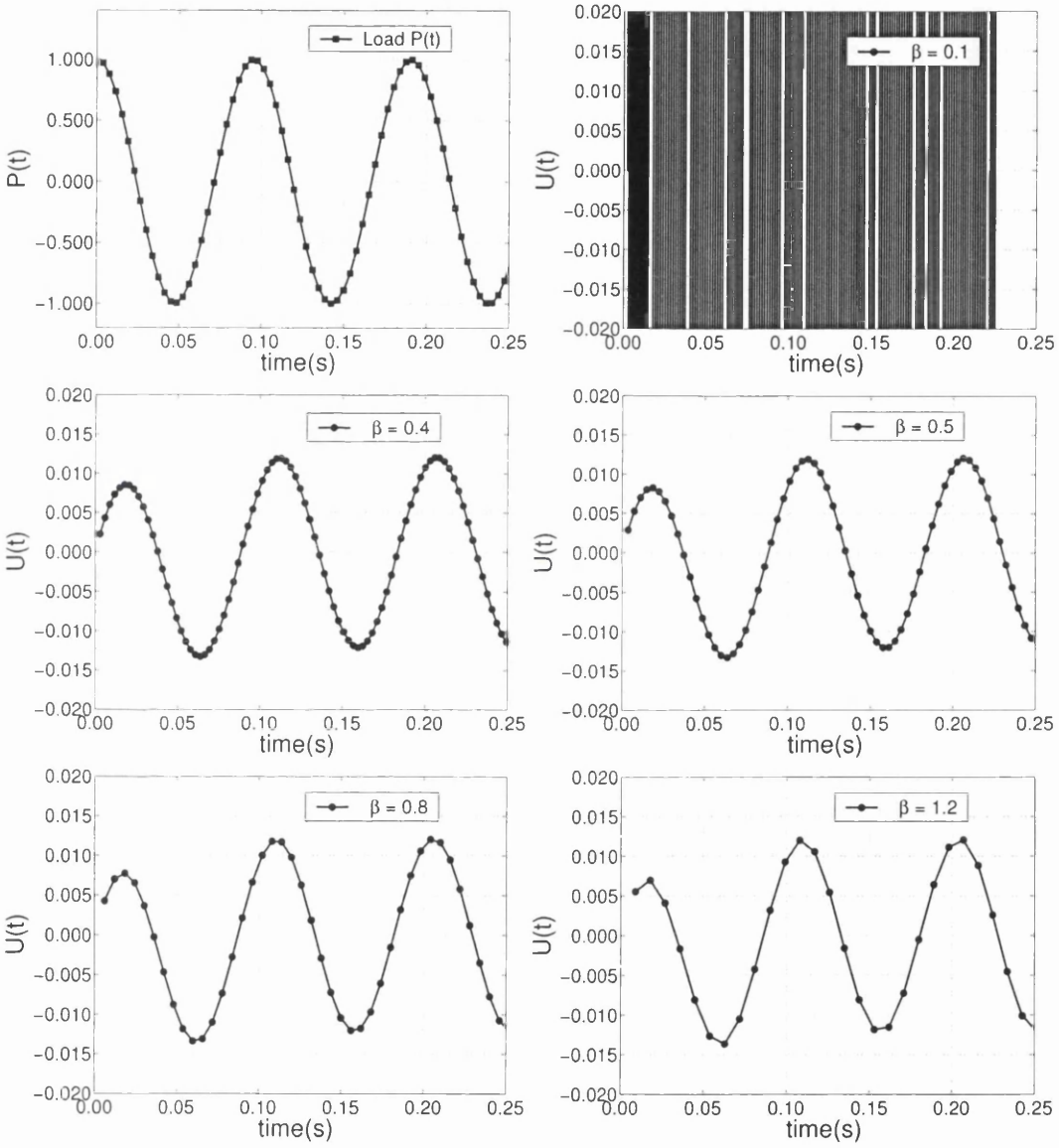


Figure 6.25: Time history of the displacement of the foundation, after refinement applied, due to harmonic loads with $\beta = 0.1,\ 0.3,\ 0.5,\ 0.8,\ 1.2$

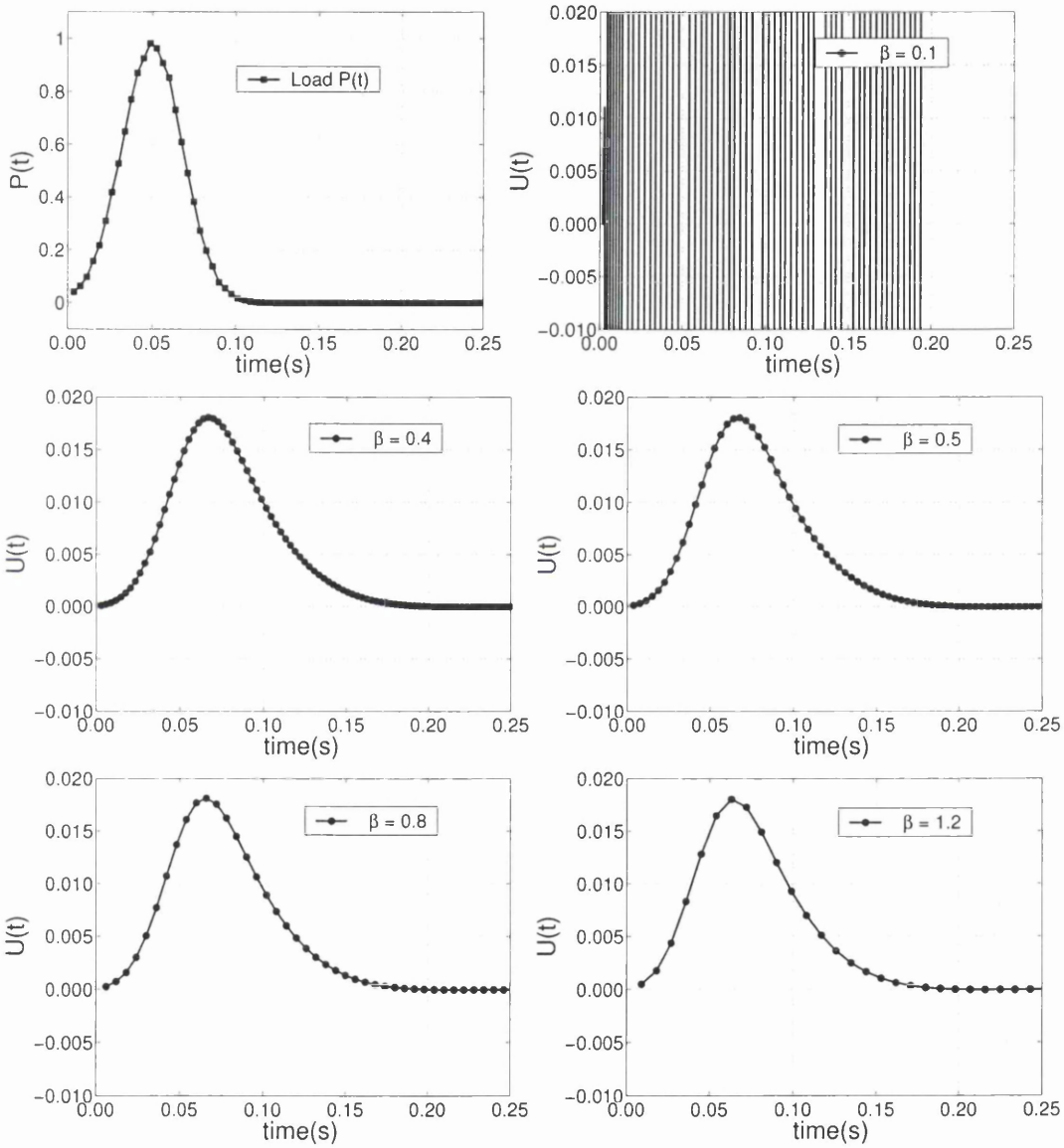
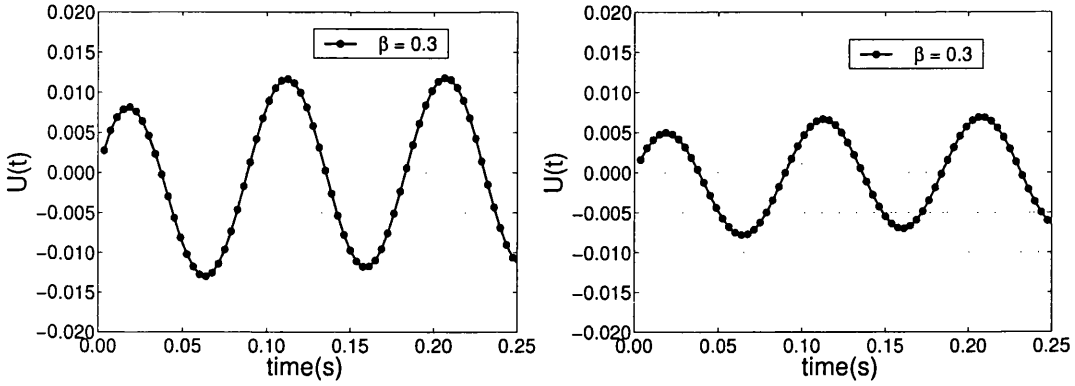


Figure 6.26: Time history of the displacement of the foundation, after refinement applied, due to impulse loads with $\beta = 0.1,\ 0.3,\ 0.5,\ 0.8,\ 1.2$

	$\beta = 0.1$	$\beta = 0.4$	$\beta = 0.5$	$\beta = 0.8$	$\beta = 1.2$
Conventional BEM (s)	N/A	142.4	89.4	53.5	41.0
adaptive BEM (s)	N/A	109.6	63.8	31.5	22.8

Table 6.1: The computing time of the rigid foundation simulations with various β values



(a) (b)

Figure 6.27: Comparison of history of vertical displacement between foundations of two sizes (a) $2b = 15m$ (b) $2b = 20m$

BEM, the sizes of the elements change at each time step. Thus, the final maximum time-space ratio β is used for the adaptive case. When β =0.4, the adaptive BEM is 23.2% faster than the conventional BEM. When the β values increase (that is, the initial mesh grows more coarse), the computing efficiency of adaptive BEM increases from 23.2% to 44.4%.

In case (c), we compute the dynamic response of a larger foundation. The time histories of vertical displacement of the two foundations is shown in Fig. 6.27. Naturally, the amplitude of displacement of the larger foundation is smaller, but otherwise there is very little difference in the results.

In order to compare with numerical results obtained by other researchers, the vertical compliance of the rigid foundation is expressed as a function of the dimensionless frequency $a_0 = \omega b/c_2$. A series of unit amplitude sinusoidal loads of various frequencies are applied to the rigid foundation, and the time history of the response is calculated. Then, after several cycle of oscillation, the system reaches its steady state, as shown in Fig. 6.22 and Fig. 6.25. The peak value $\triangle_z$ of the vertical displacement is used to compute the vertical compliance:

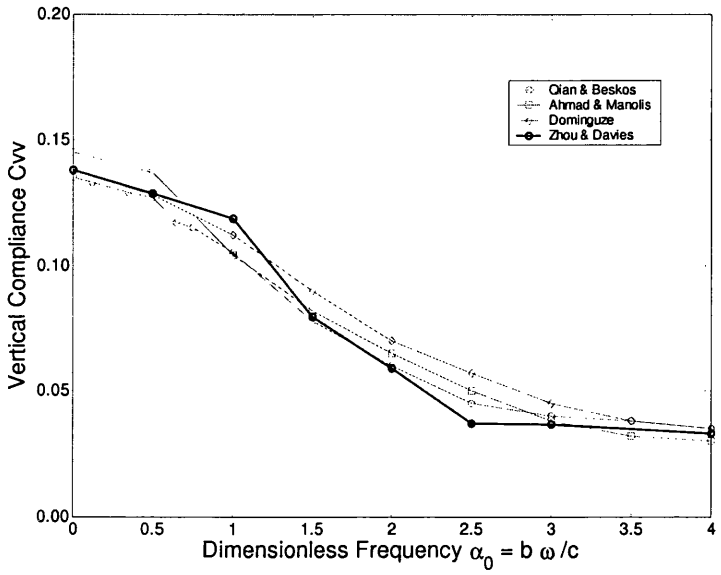


Figure 6.28: Vertical compliance of a square rigid foundation

$$C_{vv} = \mu b \triangle_z \tag{6.59}$$

The computational results of vertical compliance for the adaptive scheme are shown in Fig. 6.28, and compared with results from other researchers. These are: (a) Qian & Beskos (1995) [84] who used 8×8 quadratic elements in the frequency-domain; (b) Ahmad & Manolis (1988) [2] who used 18 quadratic elements assuming quadrennial symmetry in the frequency-domain; and (c) Dominguez (1978) [20] who used an 8×8 constant element discretization in the frequency-domain. It is obvious from Fig. 6.28 that the time-domain approach yields results in reasonable agreement with the three frequency-domain approaches.

From this numerical example, we can draw following conclusions:

- The dynamic displacement of a massless rigid foundation decreases at higher loading frequencies. Displacements are phase-shifted with respect to the input harmonic load.

- The time-space ratio β needs to be chosen with care. In this case, a value between 0.4 and 0.8 is proper one. Smaller values of β leads to instability

while bigger ones leads to inaccuracy.

- The refinement scheme improves accuracy for almost all cases, except when β is very small. It also make the computation more efficient.
- To reach the similar accuracy, the adaptive BEM is about $20\% \sim 45\%$ faster than the conventional BEM.

6.5.3 Spherical cavity embedded in a half-space

A spherical cavity is embedded in a half-space is shown in Fig. 6.29. We wish to explore its dynamic response to explosive pressure at various embedding depths. In particular, we examines:

- The time history of the displacements of the cavity wall under various explosive loads (Heaviside, Gaussian etc.);
- The influence of embedment depth.
- The influence of different discretization of the free ground surface.
- The influence on stability and accuracy of element size , time-space ratios β and refinement scheme.

Here $R = 1.00m$ is the radius of the spherical cavity, and d is the distance between the cavity center and the ground surface. The numerical test program involves:

The radial pressure $p_0 = 6.90 \times 10^6 Pa$ is suddenly applied and maintained on the sphere wall. Two embedment depths are examined. We compute the time history of the displacements at the cavity wall and the ground surface on points A, B, C, D (as shown in Fig. 6.30) for embedment depth $d = 2R$, $5R$. the influence of ground surface discretization is explored by using two meshes: one of dimension is $12R \times 12R$ and one of dimension is $30R \times 30R$. A Gaussian-type impulse load is also considered. We also capture the snapshots of the ground surface's displacement movements.

We mesh the cavity and the free surface with 199 nodes and 92 triangular elements, as shown in Fig. 6.31. There are 56 triangular elements for the cavity: each

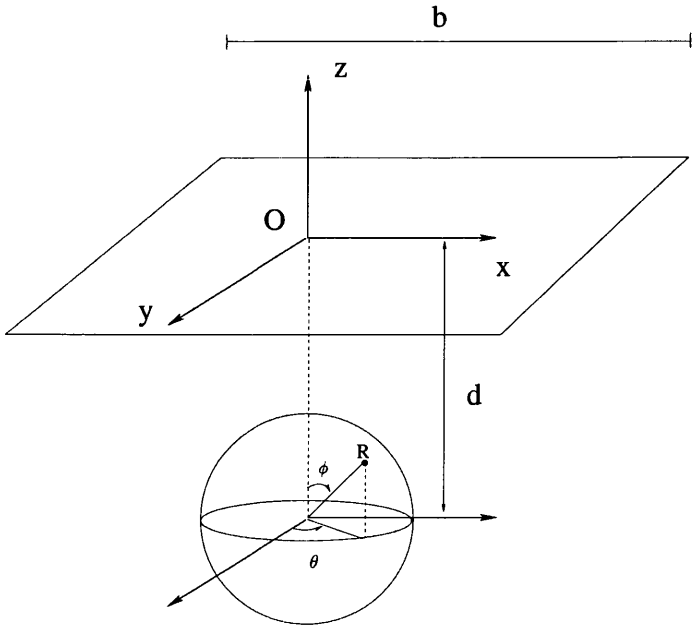


Figure 6.29: A spherical cavity embedded in a half-space

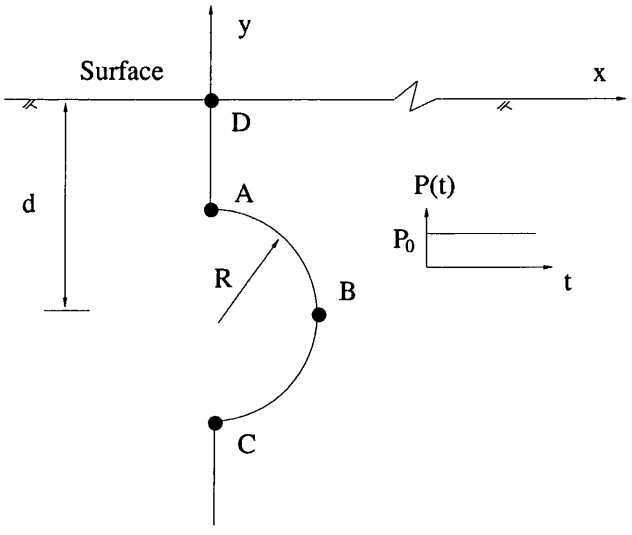


Figure 6.30: The configuration of spherical cavity embedded in a half-space

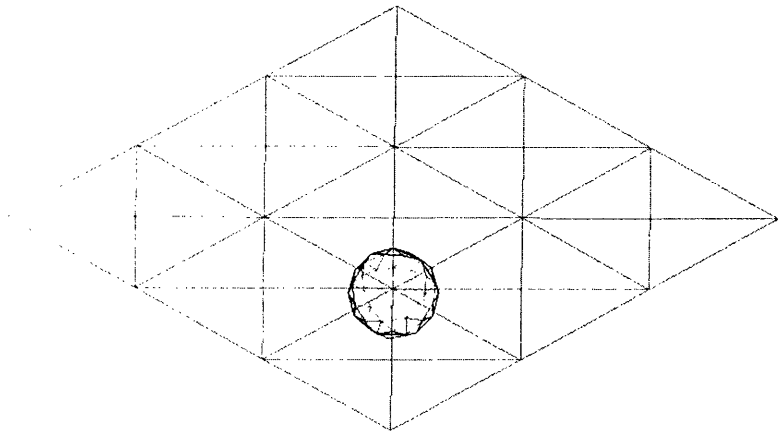


Figure 6.31: Mesh of underground sphere and free ground surface with b = 6 m

element is about $dh = 0.5m$ in size. The free surface is represented by 36 elements, as shown. Each of these element is about $dh = 2.0m$ in size. We compute the time history of the displacement of the cavity wall and surface on points A, B, C, D, for depth (d) of 2R and 5R as shown in Fig. 6.32 and Fig. 6.33. The *'reference'* curve in these figures represents the dynamic response of a cavity embedded in an infinite domain.

It is clear from Fig. 6.32 and Fig. 6.33 that dynamic responses are larger than that in reference case, which reflects the fact that the static displacements are also greater. The surface effect also results in non-symmetry (compare results fro points A, B, & C). The influence of the embedment depth of the cavity is also quite evident. When D = 2R, the displacement at the surface point D jumps to a value which is even greater than those on the cavity wall. However, when D = 5R, the surface displacement is relatively small. Because a non-uniform mesh is used, β is not the same for all elements. The larger value β_{max} is a control parameter in this case, which means that the smaller mesh dictates the overall accuracy of the solution. The time-space ratio β_{max} does not appear to affect the stability too much is this case. Good results are obtained for values of β_{max} in the range 0.3-0.8.

We now enlarge the ground surface mesh to $30R \times 30R$, using total of 335 nodes and 156 triangular elements, as shown in Fig. 6.34. There are 56 triangular elements for the cavity and 100 elements for the free surface. We compute the time history

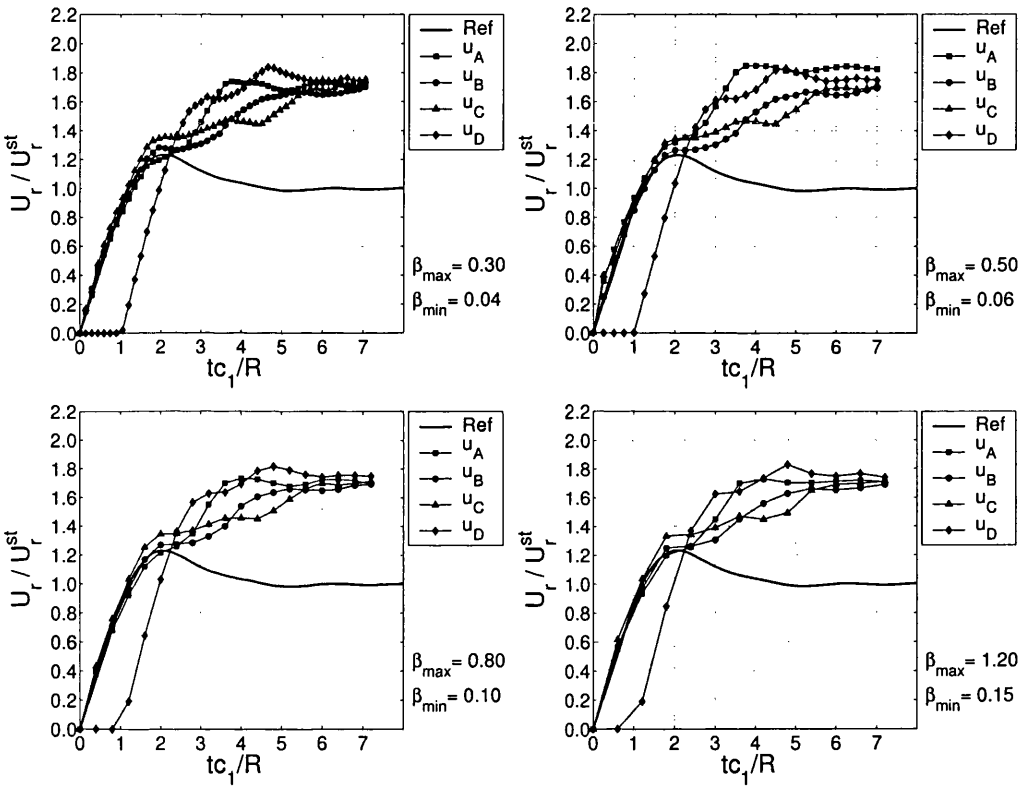


Figure 6.32: Time history of the displacement of the cavity wall and surface on points A, B, C, D due to Heaviside load with $\beta_{max} = 0.3,\ 0.5,\ 0.8,\ 1.2$; depth = 2R, b = 6m

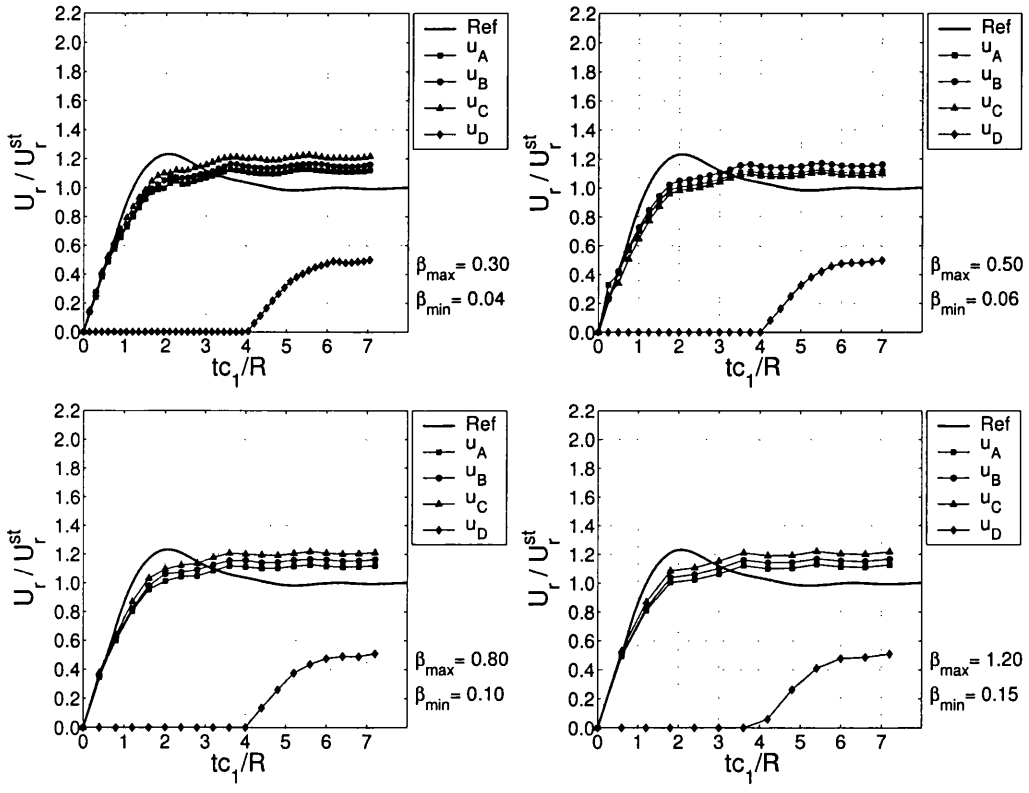


Figure 6.33: Time history of the displacement of the cavity wall and surface on points A, B, C, D due to Heaviside load with $\beta_{max} = 0.3,\ 0.5,\ 0.8,\ 1.2$; depth = 5R, b = 6 m

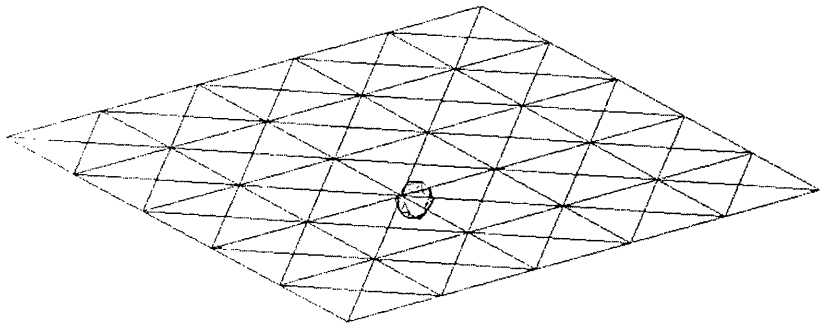


Figure 6.34: Mesh of underground sphere and free ground surface with b = 15 m

of the displacements at points A, B, C, D, with two different depths $d = 2R$, $5R$, as shown in Fig. 6.35 and Fig. 6.36.

The results are very similar to those obtained earlier: slightly greater displacement are observed at later times. To make it more obvious, a zoom-in version of the time history of the displacements at points D under the impulse loading is shown in Fig. 6.37. Indeed two results almost overlap with each other except the very end of the time.

Using the coarse surface mesh of $12R \times 12R$, the adaptive scheme is now examined. The time history of the displacements at points A, B, C, D are shown in Fig. 6.38 and Fig. 6.39. It is observed that the displacements of point A and point D are larger than those obtained earlier.

To compare the effects of using adaptive schemes, a zoom-in version of the time history of the displacements at points D under the impulse loading is shown in Fig. 6.40. The adaptive BEM resolves the wave front more accurately than the convention BEM.

We apply the Gaussian-type impulse load and obtain the results shown in Fig. 6.41 and Fig. 6.42. On the shallowly-embedded case, we see that the displacements of the cavity wall closely follows the impulse loading, and that ground surface suffers even greater displacements than the cavity itself. For the deeply-embedded case, there is no strong interaction between the sphere and ground and surface displacements are relatively small. Snapshots of ground surface displacement at various times are shown in Fig. 6.43. We see that a wave is formed in the center which later propagates in all directions.

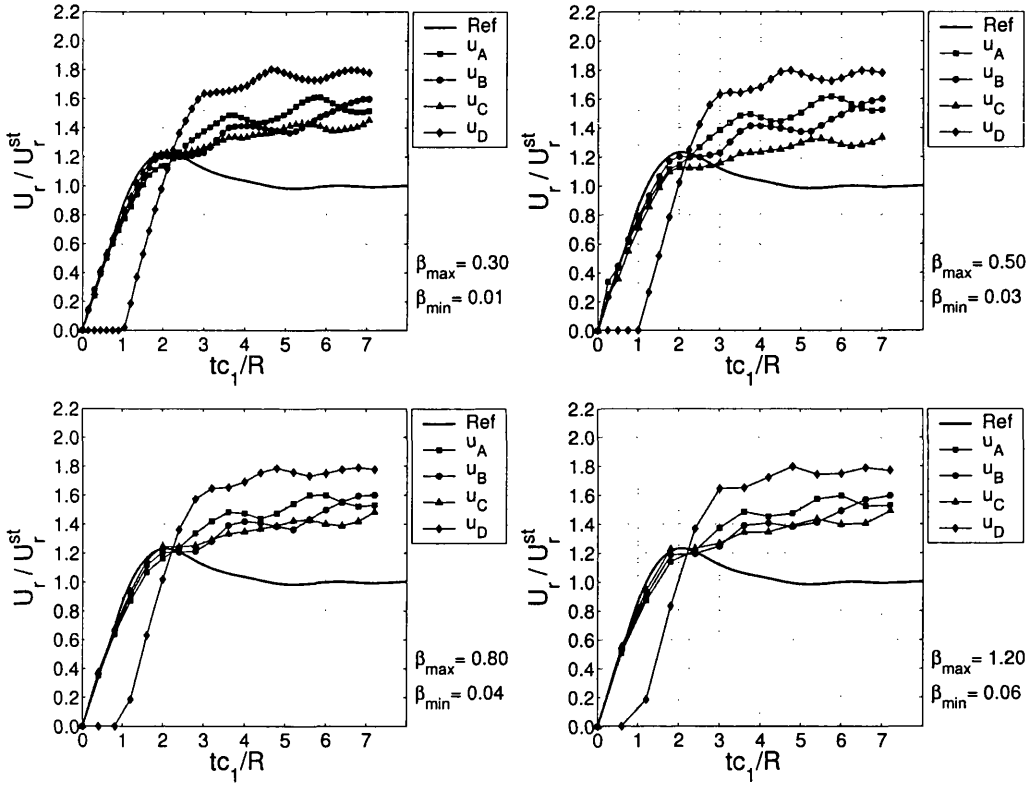


Figure 6.35: Time history of the displacement of the cavity wall and surface on points A, B, C, D due to Heaviside load with $\beta_{max} = 0.3,\ 0.5,\ 0.8,\ 1.2$; depth = 2R, b=15 m

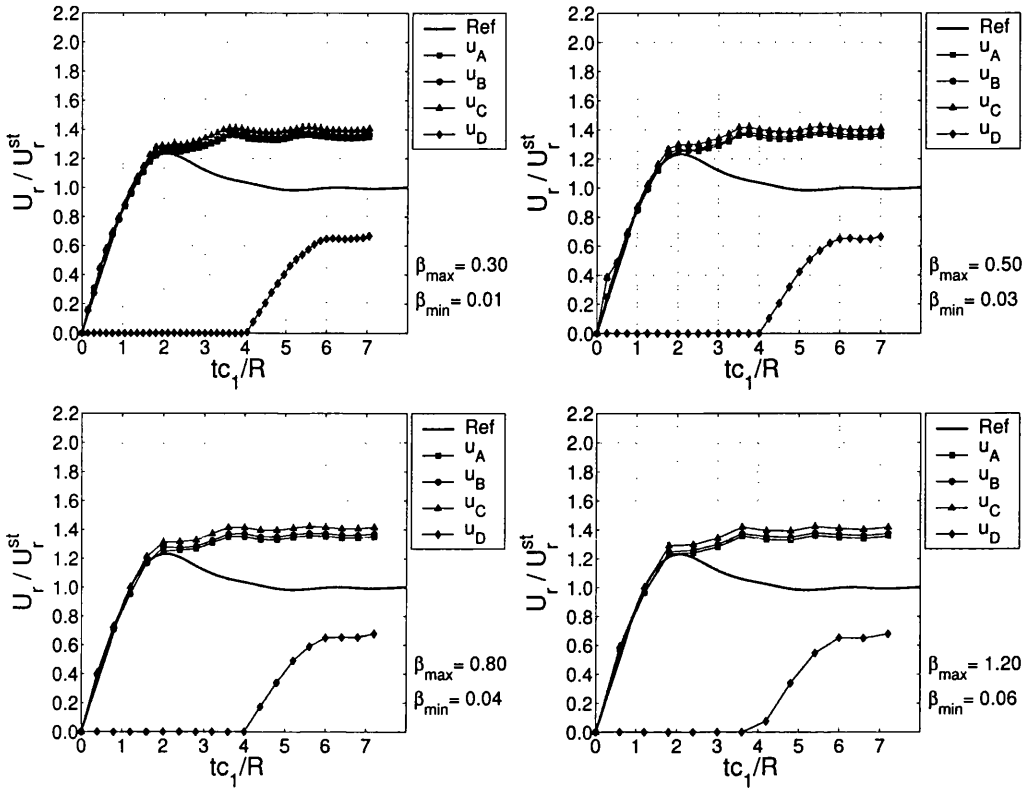


Figure 6.36: Time history of the radial displacement of the cavity wall and surface on points A, B, C, D due to Heaviside load with $\beta_{max} = 0.3,\ 0.5,\ 0.8,\ 1.2$; depth = 5R, b = 15 m

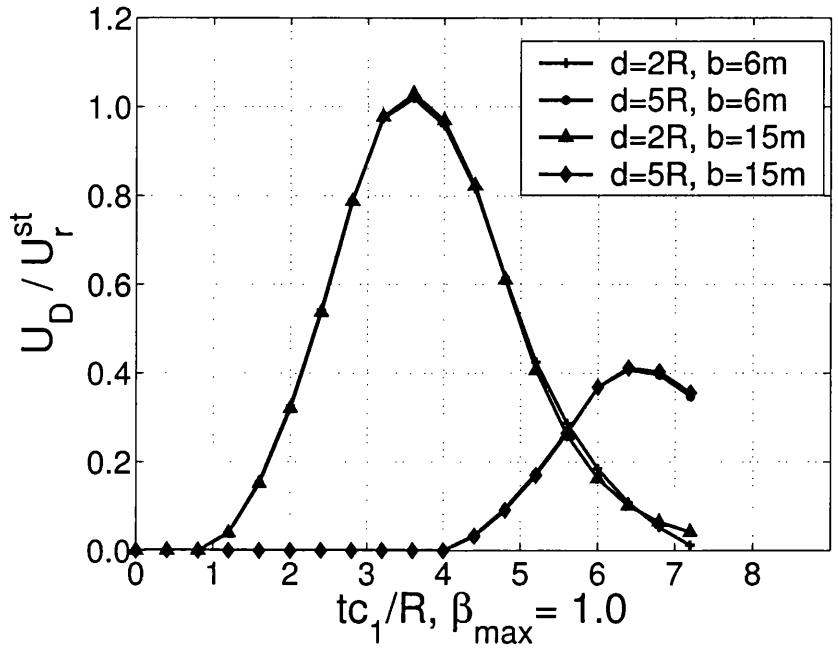


Figure 6.37: Radial displacement history at point D of the cavity at depth of 2R and 5R, with the different sizes of the free surface mesh

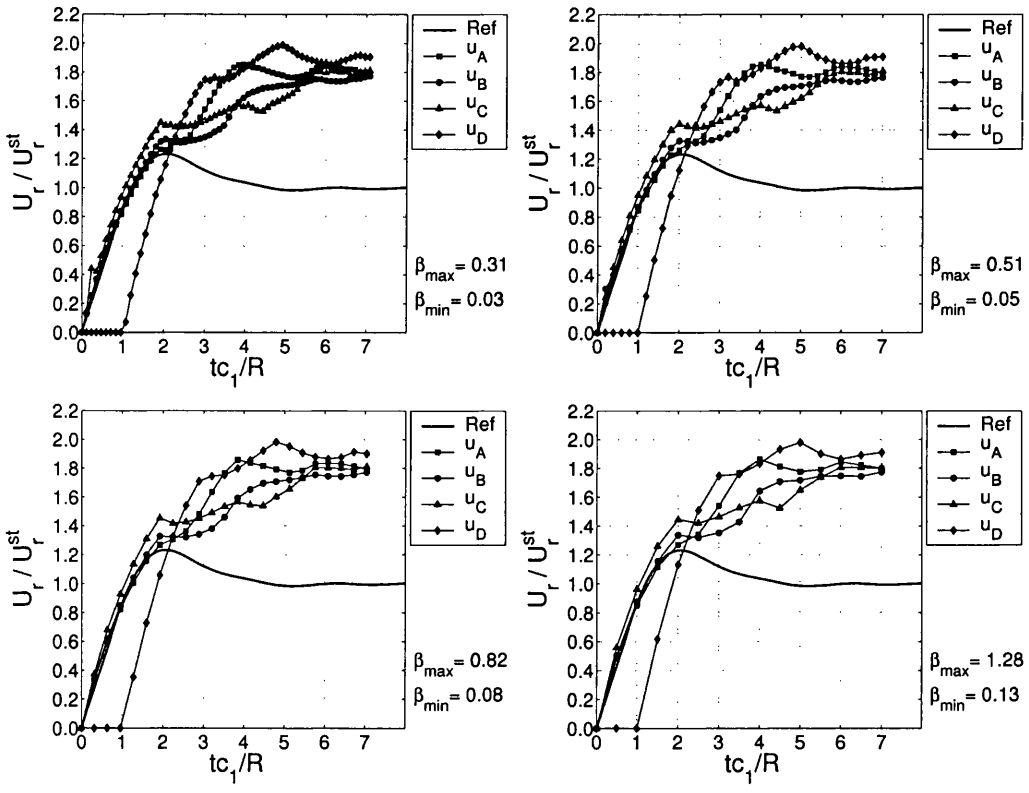


Figure 6.38: Time history of the radial displacement of the s wall and surface on points A, B, C, D due to Heaviside load with $\beta_{max} = 0.31,\ 0.51,\ 0.82,\ 1.29$, depth = 2R, b = 6m, refined scheme

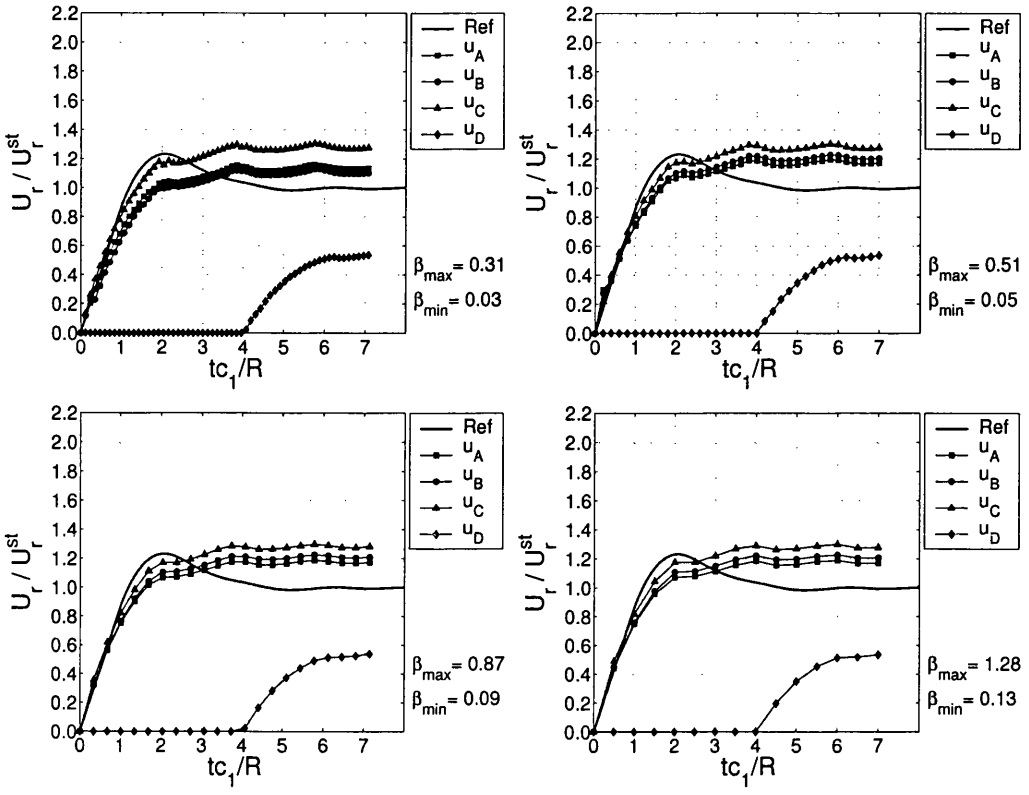


Figure 6.39: Time history of the radial displacement of the sphere wall and surface on points A, B, C, D due to Heaviside load with $\beta_{max} = 0.31,\ 0.51,\ 0.87,\ 1.29$, depth = 5R, b = 6 m, refined scheme

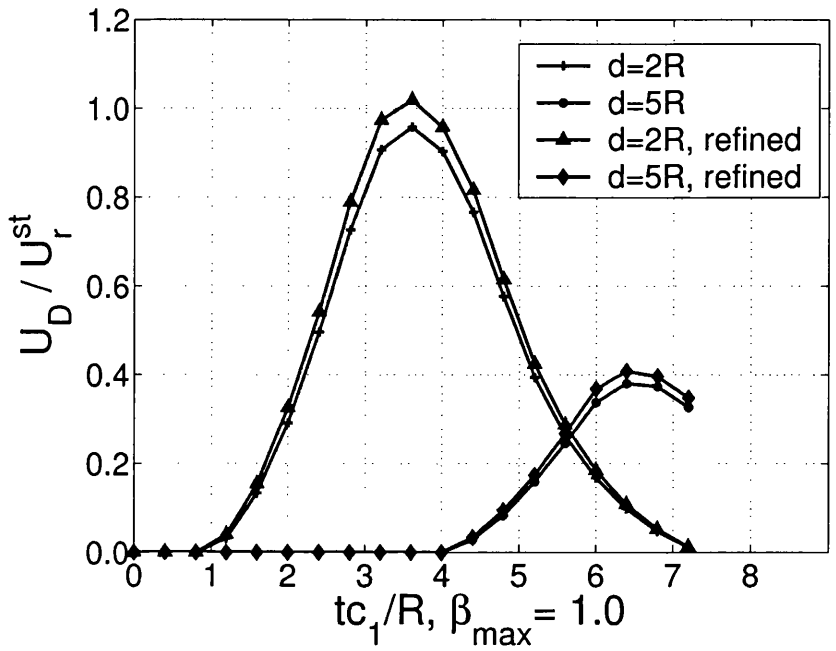


Figure 6.40: Radial displacement history at point D of the cavity at depth of 2R and 5R, with and without the refinement

	$\beta = 0.3$	$\beta = 0.5$	$\beta = 0.9$	$\beta = 1.3$	$\beta = 2.0$
Conventional BEM (s)	116.0	72.5	42.8	33.2	19.9
adaptive BEM (s)	89.3	51.8	25.2	18.4	11.04

Table 6.2: The computing time of the underground cavity simulations with various β values

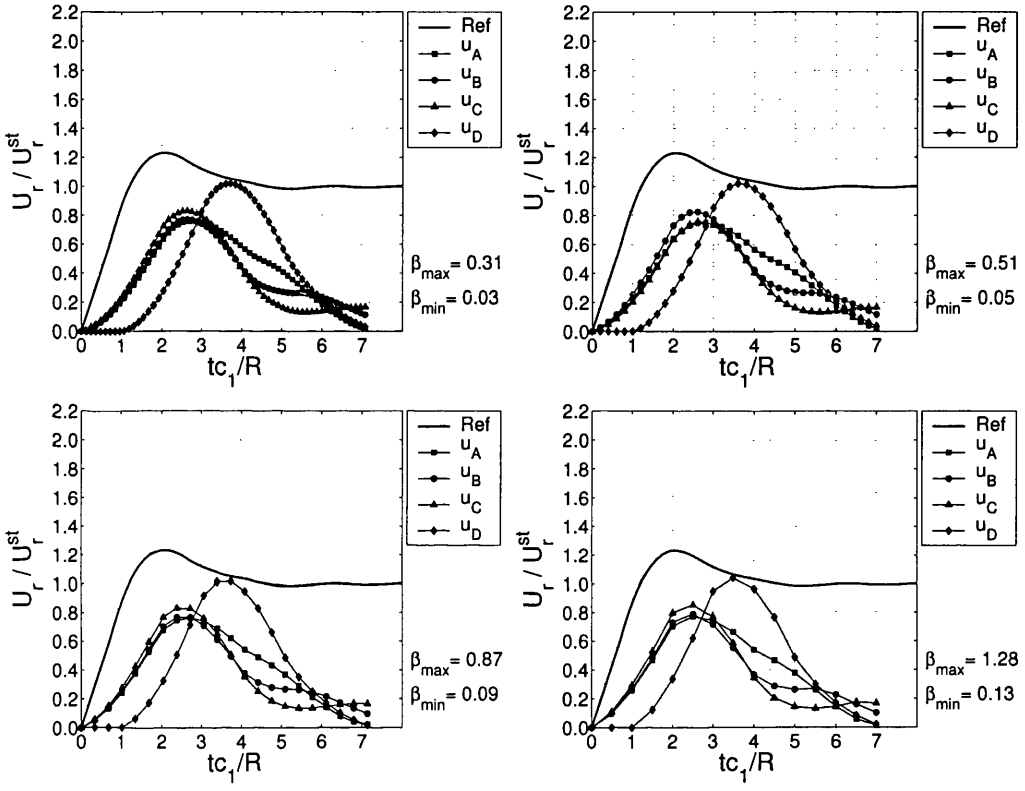


Figure 6.41: Time history of the displacement of the cavity wall and surface on points A, B, C, D due to Gaussian-type impulse load with β_{max} = 0.31, 0.51, 0.82, 1.29 , depth = 2R, b = 6m, refined scheme

To obtain the efficiency of adaptive BEM, we record the computing time for each simulation with the time-space ratio β from 0.3 to 2.0 (Table 6.2). For the adaptive BEM, the sizes of the elements change at each time step. Thus, the final maximum time-space ratio β is used for the adaptive case. When β =0.3, the adaptive BEM is 23.0% faster than the conventional BEM. When the β values increase (that is, the initial mesh grows more coarse), the computing efficiency of adaptive BEM increases from 23.0% to 44.7%.

From this numerical example, we can draw following conclusions:

- The dynamic response of the near-surface cavity subjected to Heaviside loading

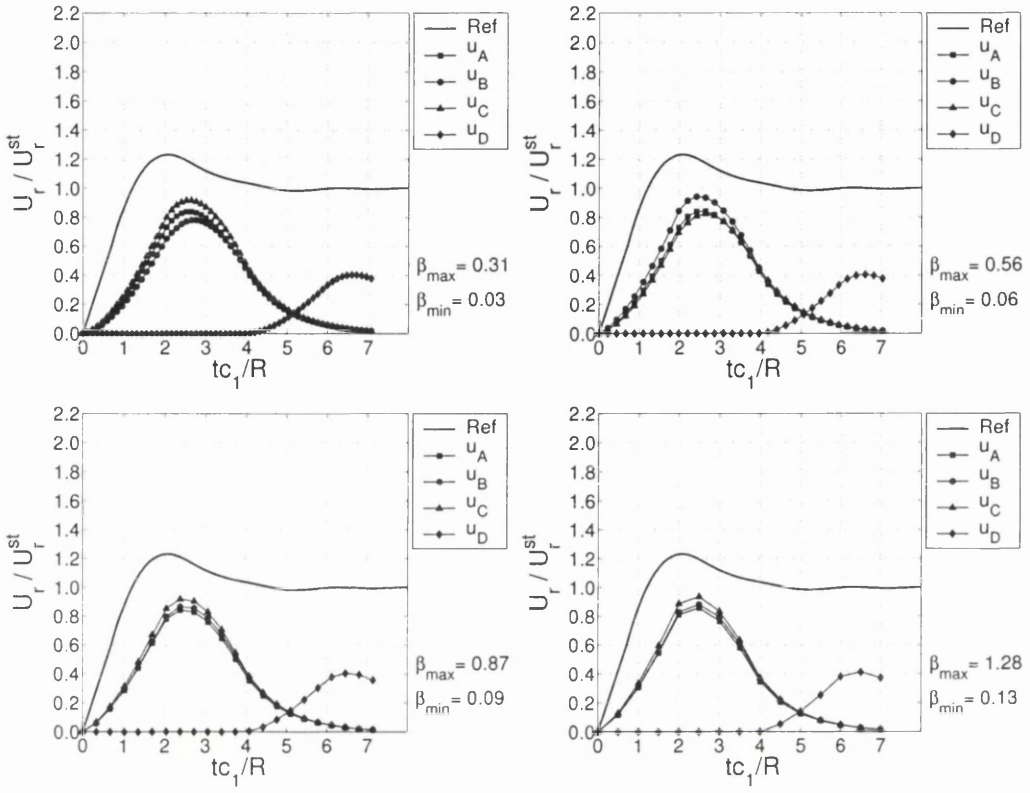


Figure 6.42: Time history of the displacement of the cavity wall and surface on points A, B, C, D due to Gaussian-type impulse load with $\beta_{max} = 0.31,\ 0.51,\ 0.87,\ 1.29$; depth = 5R, b = 6 m, refined scheme

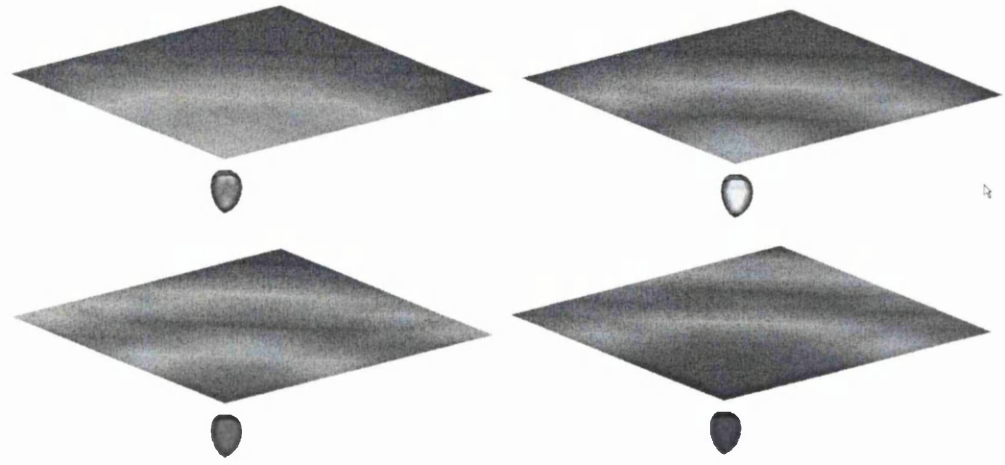


Figure 6.43: Snapshots of the whole ground surface mesh displacement at time t = 0.015s, 0.026s, 0.040, 0.065s

converges to infinite domain case as embedment depth increases.

- The time-space ration β has to be chosen with care. In this case, a suitable choice is between 0.3 and 0.8.
- The extent of the ground surface mesh has a little influence on response to impulse loads, except at later times.
- To reach the similar accuracy, the adaptive BEM is about 23% $\sim$ 45% faster than the conventional BEM.

6.6 Summary

Based on the BEM integral equation for 3D elastodynamic problems, gradient-based and resolution-based error indicators are used to locate high gradient boundaries. A space refinement scheme and local time stepping is employed to capture the moving wave fronts. We apply the method to solve the problems of a 3D spherical cavity under explosive loading, the simulation of a rigid foundation in a half-space under the periodic loading, and a underground cavity under explosive loading. The adaptive BEM solver can solve the problem with a coarse mesh. Compared with conventional dynamic BEM, the same accuracy is achieved with fewer nodes and less CPU time.

Chapter 7

Parallel Programming

7.1 Introduction

Dynamic problems are computationally intensive. Data must be computed for many time steps and new influence matrices are computed at each time step. There are also substantial memory demands because data for past time steps has to be retained. To obtain more efficient solutions, parallelization of dynamic BEM is very attractive.

For an efficient parallel implementation of dynamic BEM, the most time-consuming computational procedures, such as the time stepping schemes, numerical integrals etc., must be analyzed for their feasibility for parallelization. Section 7.2 describes parallel algorithms of computing influence matrices, time stepping schemes, numerical integrals and iterative solvers for dense and non-symmetric system. The implementation of these algorithms is described in Section 7.3. Finally, some examples of applications of parallel space-time BEM are discussed in Section 7.4.

7.2 Parallel algorithm for the dynamic BEM

7.2.1 Basic strategy of parallelization for BEM

The main strategy in parallelization is to partition the overall problem into separate tasks; to allocate tasks to different processors and to synchronize those tasks to obtain final results. Parallel programming can only be applied to tasks that are

inherently parallelizable, which implies that the tasks are data independent. Or, simply put, each task can run independently without requiring data from other tasks. A problem can be partitioned based on domain decomposition or functional decomposition, or a combination of these two. There are two major approaches to parallel programming.

- Implicit parallelism — the system (the operating system, the compiler, or some other programs) partitions the problem and allocates tasks to processors automatically. For example, current dual-CPU Microsoft Windows OS can function like this.
- Explicit parallelism — the programmer must annotate the program to show how the problem is to be partitioned.

Many computational procedures in dynamic BEM are data independent and thus inherently parallelizable. Examples are computing the influence matrices $[G]^{nm}$, $[H]^{nm}$ at different time steps (n — *current time step,* m — *all past time steps*), that is, computing the entries G_{ij}, H_{ij} in the influence matrices. At a small scale, each function evaluation at the Gauss points for the numerical integrals can be performed in parallel. The following sections describe these processes in more detail.

7.2.2 Computing influence matrices

The boundary integral equation for the dynamic BEM is:

$$\begin{aligned} c_{lk}^{i} u_{k}^{i}\left(\boldsymbol{x}^{i}, t\right) &= \int_{S} u_{lk}^{*}\left(\boldsymbol{x}, t-\tau; \boldsymbol{x}^{i}\right) * p_{k}\left(\boldsymbol{x}, \tau\right) dS\left(\boldsymbol{x}, \tau\right) \\ &- \int_{S} p_{lk}^{*}\left(\boldsymbol{x}, t-\tau; \boldsymbol{x}^{i}\right) * u_{k}(\boldsymbol{x}, \tau) dS(\boldsymbol{x}, \tau) \end{aligned} \tag{7.1}$$

where $\boldsymbol{x}^i$ refers to a source point; $\boldsymbol{x}$ refers to a field point on the boundary S; u_{lk}^* is the Green's function for displacements; p_{lk}^* is the Green's function for tractions; u_k and p_k are displacements and tractions on the boundary.

Its discrete form is:

$$
\begin{aligned}
c_{lk}^{i} u_{k}^{i} &= \sum_{j=1}^{N} \Big(\int_{S_j} u_{lk}^{*} \cdot N^{j}(\xi,\eta,\tau)\, p_{k}^{j} dS(\boldsymbol{x},\tau) \\
&- \int_{S_j} p_{lk}^{*} \cdot N^{j}(\xi,\eta,\tau)\, u_{k}^{j} dS(\boldsymbol{x},\tau)
\end{aligned} \tag{7.2}
$$

where u_k^j, p_k^j are nodal values of displacements and tractions; $N^j(\xi,\eta,\tau)$ are shape functions; S_j is the surface boundary element, and$N$ is the total number of boundary elements.

The influence matrices are computed from:

$$
\begin{aligned}
G_{lk}^{ij} &= \int_{S_j} u_{lk}^{*} \cdot N^{j}(\xi,\eta,\tau)\, dS(\boldsymbol{x},\tau) \\
\hat{H}_{lk}^{ij} &= \int_{S_j} p_{lk}^{*} \cdot N^{j}(\xi,\eta,\tau)\, dS(\boldsymbol{x},\tau)
\end{aligned} \tag{7.3}
$$

Clearly, the computation of these matrix entries of $[G]$ and $[H]$ is an independent process and is parallelizable. Thus, if there are N vertexes in the loop over all elements and M processors available, the N -vertex loop is partitioned into M tasks. Each processor will perform N/M vertex loops to compute the matrix entries G_{ij}, H_{ij} until the whole of the $[G]$ and $[H]$ matrices are formed. The process is shown in Fig. 7.1.

7.2.3 Parallel time stepping

This parallelization is based on domain decomposition in the time dimension. From previous chapters, the numerical methods constructed from these space-time boundary integral equations are global in time, i.e., it is necessary to compute solutions for all past time steps to obtain the current solution. The system is solved step-by-step: once $\mathbf{u}$ and $\mathbf{p}$ are known for the previous time steps, the solution at the *nth* time step is obtained from:

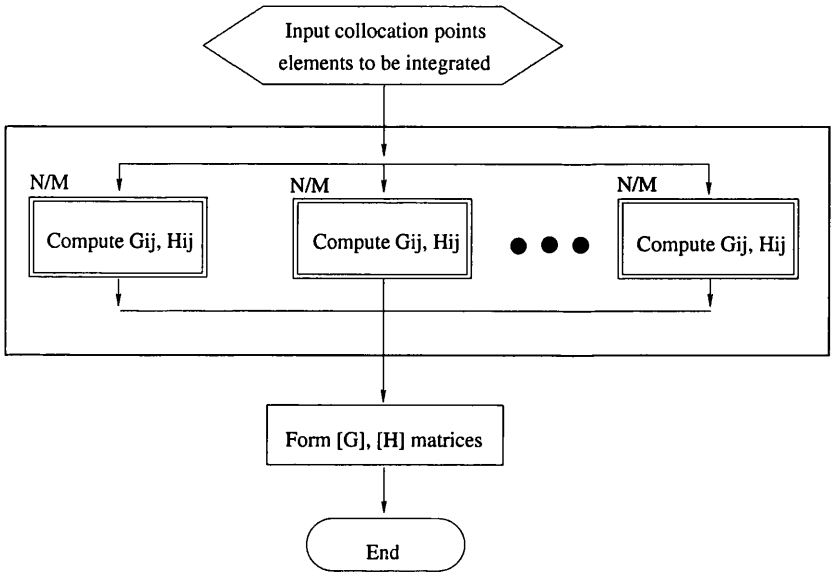


Figure 7.1: Parallel computation of matrix entries G_{ij}, H_{ij}

$$\mathbf{H}_{lk}^{(nn)ij}\mathbf{u}_k^{(n)j} = \mathbf{G}_{lk}^{(nn)ij}\mathbf{p}_k^{(n)j} + \sum_{m=1}^{n-1}\mathbf{G}_{lk}^{(nm)ij}\mathbf{p}_k^{(m)j} - \mathbf{H}_{lk}^{(nm)ij}\mathbf{u}_k^{(m)j} \tag{7.4}$$

The solution at the nth time step is dependent on the previous solution $m(m = 1, 2, ..., n-1)$. However, the $\mathbf{G}_{lk}^{(nm)ij}, \mathbf{H}_{lk}^{(nm)ij}$matrices at different time steps are independent for each other, because they are the integrals of different kernel function $U^{nm}$and P^{nm}:

$$\begin{aligned} G_{lk}^{(nm)ij} &= \int_{\Gamma_j} u_{lk}^{*} \cdot N^j(\xi,\eta,\tau)\, d\Gamma_j(\boldsymbol{x},\tau) \\ &= \int_{\Gamma_j} U^{nm}(\mathbf{x}(\xi,\eta)) \cdot N^j(\xi,\eta)\, J(\xi,\eta)\, d\xi d\eta \end{aligned}$$

$$\begin{aligned} H_{lk}^{(nm)ij} &= \int_{\Gamma_j} p_{lk}^{*} \cdot N^j(\xi,\eta,\tau)\, d\Gamma_j(\boldsymbol{x},\tau) \\ &= \int_{\Gamma_j} P^{nm}(\mathbf{x}(\xi,\eta)) \cdot N^j(\xi,\eta)\, J(\xi,\eta)\, d\xi d\eta \end{aligned} \tag{7.5}$$

where $U^{nm}(\mathbf{x}(\xi,\eta))$, $P^{nm}(\mathbf{x}(\xi,\eta))$ are time integrals of the displacement and traction kernel functions, $N^j(\xi_k,\eta_k)$ are the shape functions of 6-noded curved tri-

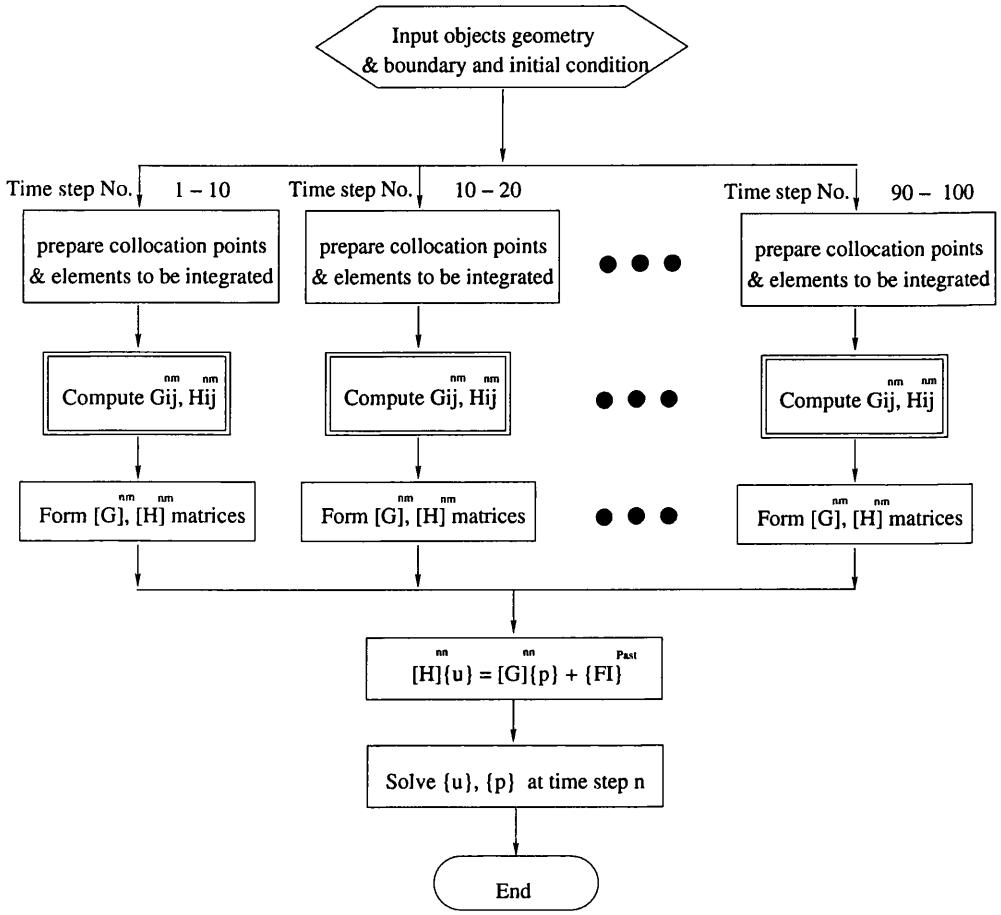


Figure 7.2: Parallel algorithm for time stepping

angular elements, $J(\xi_k, \eta_k)$ is the Jacobian for the local coordinates, and (ξ, η) are the local coordinates for the space integration.

These matrix entries are independent from each other, and can be partitioned into parallel tasks in such a way that different processors are responsible for the numerical integration of $\mathbf{G}_{lk}^{(nm)ij}, \mathbf{H}_{lk}^{(nm)ij}$ for different time steps. For example, in the schematic description of the parallel strategy in Fig. 7.2, time steps $No.1 - 10$ are assigned to task 1, time steps $No.11 - 20$ are assigned to task 2, and so on.

7.2.4 Parallel numerical integrations

Since computing function values at many Gauss points in the numerical integration is very time-consuming, it is highly desirable to parallelize it. Individual matrix coefficients for the boundary integral equations consists of integrals which are two dimensional in space and one dimensional in time. For example, the numerical

integral of the displacement kernel in 3D elastodynamics is:

$$\begin{aligned} G_{lk}^{ij} &= \int_{\Gamma_j} u_{lk}^{*} \cdot N^j(\xi,\eta,\tau)\, d\Gamma_j(\boldsymbol{x},\tau) \\ &= \int_{-1}^{1}\int_{-1}^{1} U^{nm}(\mathbf{x}(\xi,\eta)) \cdot N^j(\xi,\eta)\, J(\xi,\eta)\, d\xi d\eta \\ &= \sum_{k=1}^{K} U^{nm}(\mathbf{x}(\xi_k,\eta_k))\, N^j(\xi_k,\eta_k)\, J(\xi_k,\eta_k)\, w_k \end{aligned} \tag{7.6}$$

where $U^{nm}(\mathbf{x}(\xi,\eta))$ is the integral of the displacement kernel over the time dimension, which is done analytically; $N^j(\xi,\eta)$ are the shape functions for the space interpolation; (ξ_k,η_k) are the Gauss points; w_k are the associated quadrature weights, and K is the total number of Gauss points.

The spatial integration is approximated by Gauss quadrature. The same process is repeated for every matrix coefficient in $[G]$ and $[H]$. In order to parallelize this process, the summation series from $k = 1 - K$ is partitioned into parallel tasks. To achieve the best parallel efficiency and minimum load imbalance, the tasks taken by each processor are evenly distributed. After these tasks are completed by individual processors, the data are transmitted to a *master* processor, which calculates the final results, which is entered into $[G]$ and $[H]$. A *synchronization process* is needed to ensure that this is done correctly. The procedure is repeated for each entry until the whole of the $[G]$ and $[H]$ matrices are formed. As an example, shown in Fig. 7.3, a 16-point numerical integral scheme is run on a 4-processor parallel computer. After identifying the space coordinates of the collocation point and the Gauss points, data for four Gauss points are sent to each processor; the *synchronization process* receives the data from all processors and computes the final result.

7.2.5 Parallel iterative solvers

The solution of the linear system equations can also be parallelized. Solving these equations is usually the second most time consuming routine in BEM codes (Cunha & Telles [50]). Highly efficient machine-specific parallel implementations of linear system solvers for non-symmetric and dense matrices already exist. For example, LAPACK [48] (Linear Algebra Package) is a high performance parallel libraries specially written for solving linear algebra problems. The ScaLAPACK (or Scalable LAPACK) library includes a subset of LAPACK routines redesigned for distributed

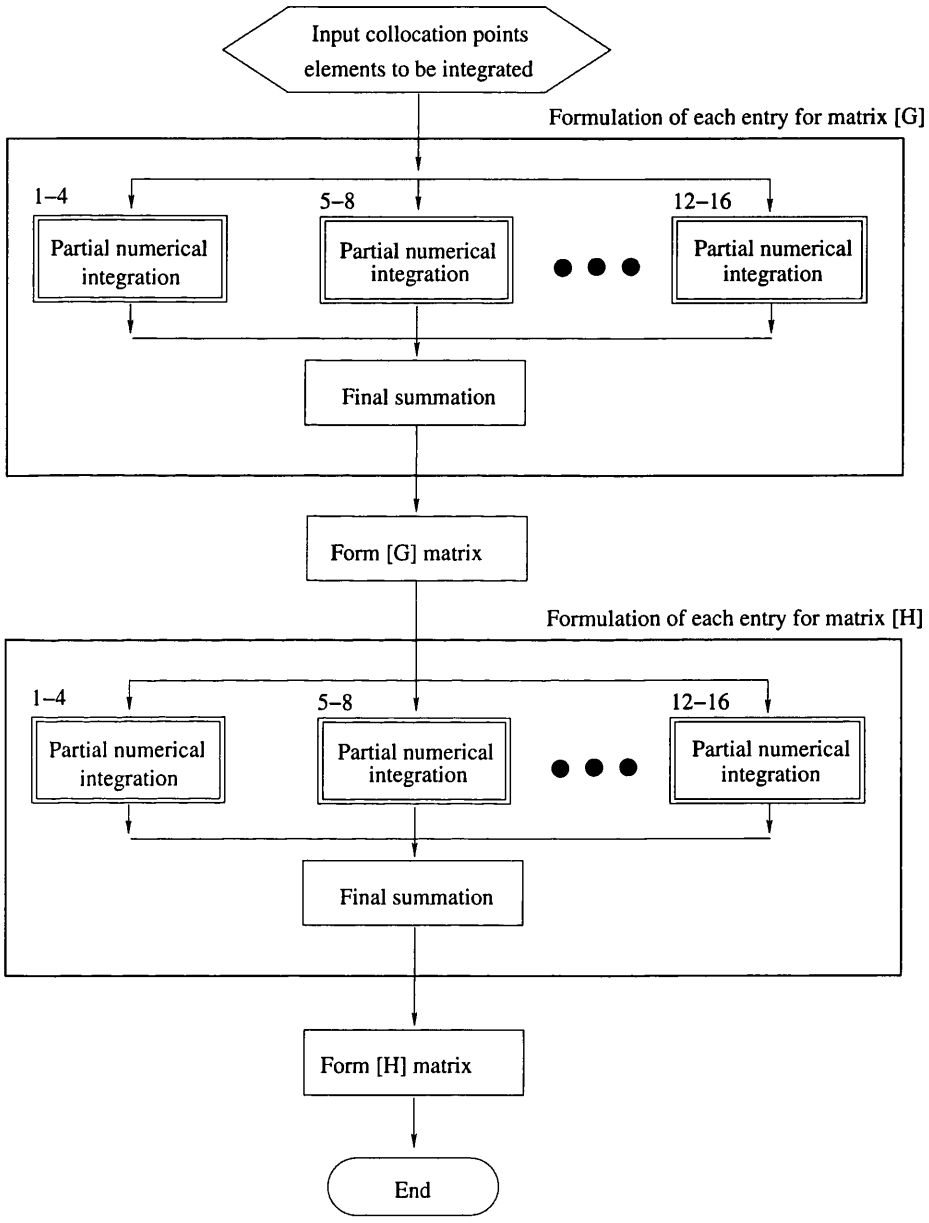


Figure 7.3: Parallel algorithm for the numerical integration of entries in $[G]$ and $[H]$ matrices

memory MIMD parallel computers (The Linux cluster which the author employed belongs to this category). ScaLAPACK is designed for heterogeneous computing and is portable on any computer that supports Message Passing Interface (MPI).

7.3 Implementation of parallel BEM

7.3.1 Parallel hardware and software architectures

Hardware environment for parallel computing

The hardware environment used here for parallel computing is a Linux cluster. A Linux cluster is a group of loosely coupled PCs. The component PCs of a cluster are commonly connected to each other through a fast local area network (LAN). They are usually controlled by a single computer called *the master processor* which coordinates and improves their performance.

A Linux cluster provides increased performance by splitting a computational task across the processors in the cluster. It is optimized for workloads which require separate processors to communicate actively during the computation. Many functions for parallelization may be realized by using software libraries such as MPI, which is specially designed for writing scientific applications on parallel platforms.

Software environment for parallel computing

Message Passing Interface (MPI) is a software environment for parallel computing. It is a computer communications protocol and is implemented via libraries which can be easily incorporated within user's programs. It is the most popular standard for communication among individual processors running a parallel program on a distributed memory system. MPI implementations consist of a library of routines that can be called from FORTRAN, C and C++ programs. The advantage of MPI over older message-passing libraries is that it is both portable and fast. MPI has been implemented for almost every distributed memory architecture, and each implementation is optimized for the hardware on which it runs.

Algorithm 9 Data structure of MPI_vertex_List

```
struct MPI_vertex_List
{
int CPUNumber;              // Number of processors available
Surface_Vertex * para_tag; // Mark the begin and end of subtask
Surface_Vertex * meshList;
}
```

Algorithm 10 Data structure of tri_elements representation of a matrix

```
struct tri_elements[n]
{
int i; // entry's row No.
int j; // entry's column No.
double Gij; // Value of Gij
double Hij; // Value of Hij
}
```

7.3.2 Implementation

To efficiently implement parallelization of the calculation of the influence matrices, some special data structures are defined to decrease the communication traffic between processors. First, we send a copy of the vertex list and other geometric data to each processor, and identify which vertexes should be processed by which processor. A header is created for this function, as shown in Algorithm 9.

Since each processor calculates only some entries of the matrices $[G]$ and $[H]$, it is a waste of communication bandwidth to use full size matrices in transmission. A more compact data structure called a *tri_ element* representation of a matrix is used here, as shown in Algorithm 10. Hence, it is a *tri_ element* array which is sent from each *slave* processor to the *master* processor.

The whole parallel process is shown in Fig. 7.4. The usual time stepping is employed for the parallel version of dynamic BEM. At each time step, after the intersection of sphere pairs and boundaries, a copy of the vertex list with the intersection subtriangles is sent to each processor. Each processor independently computes matrix entries of G_{ij}, H_{ij}, stores them in the *tri_ element* array and sends them back to the *master* processor which assembles the general matrices $[G]$ and $[H]$. When the past influence vector $\{FI\}$ is computed, a copy of the vertexes and mesh list at past

times and the time step range is sent to each processor. Each processor computes its allocated intersection, with the sphere radii defined by time step m. Then, the matrices $[G]^{nm}$ and $[H]^{nm}$and $\{FI\}^m$are computed. Finally $\{FI\}^m$ is transmitted to the *master* processor which computes the final results.

7.4 Numerical examples

3D underground explosion modeling

The dynamic response of a 3D riverbank-tunnel system (shown in Fig. 7.5) subjected to an impact load inside the tunnel is considered in this example. The tunnel center is located at a depth of $14.5m$ from the ground surface. It is modeled as a cylinder $r = 1.5m$, with one end open and the other closed. The explosion takes place in the middle of the tunnel, which is $9m$ away from the closed end and over a distance of $6m$. It is modeled as a Heaviside function suddenly applied to the tunnel wall, with radial pressure $p_0 = 6.90 \times 10^6 Pa$. The river bank is $5.5m$ wide at the top, with a slope height of $14.5m$ as shown. The density of soil is $\rho = 2.67 \times 10^6 kg/m^3$; Poisson's ratio $\nu = 0.25$.

The ground surface, the slope and the wall of the tunnel are discretized by boundary elements. By symmetry, only half of the riverbank-tunnel system is modeled. The model, using the pre-processor package *Gid,* is shown in Fig. 7.6. In the mesh, there are 1,649 triangular elements and 3,363 nodes, as shown in Fig. 7.7. It is a formidable computational problem, which demands more than 2 Gbyte memory and is beyond the computing power of ordinary desktops.

The performance of a parallel BEM can be measured in terms of two indexes, i.e. speed-up and efficiency. The speed-up index is a measurement of the improvement gained by the parallel program with regard to a single processor, defined as:

$$Speedup = \frac{user\ time\ for\ one\ processor}{user\ time\ for\ N\ processors} \tag{7.7}$$

High speed-up implies that the computational time taken will be very much reduced but it may be achieved by using a large number of processors, and therefore, the efficiency may not be high. The efficiency of the scheme can be measured by

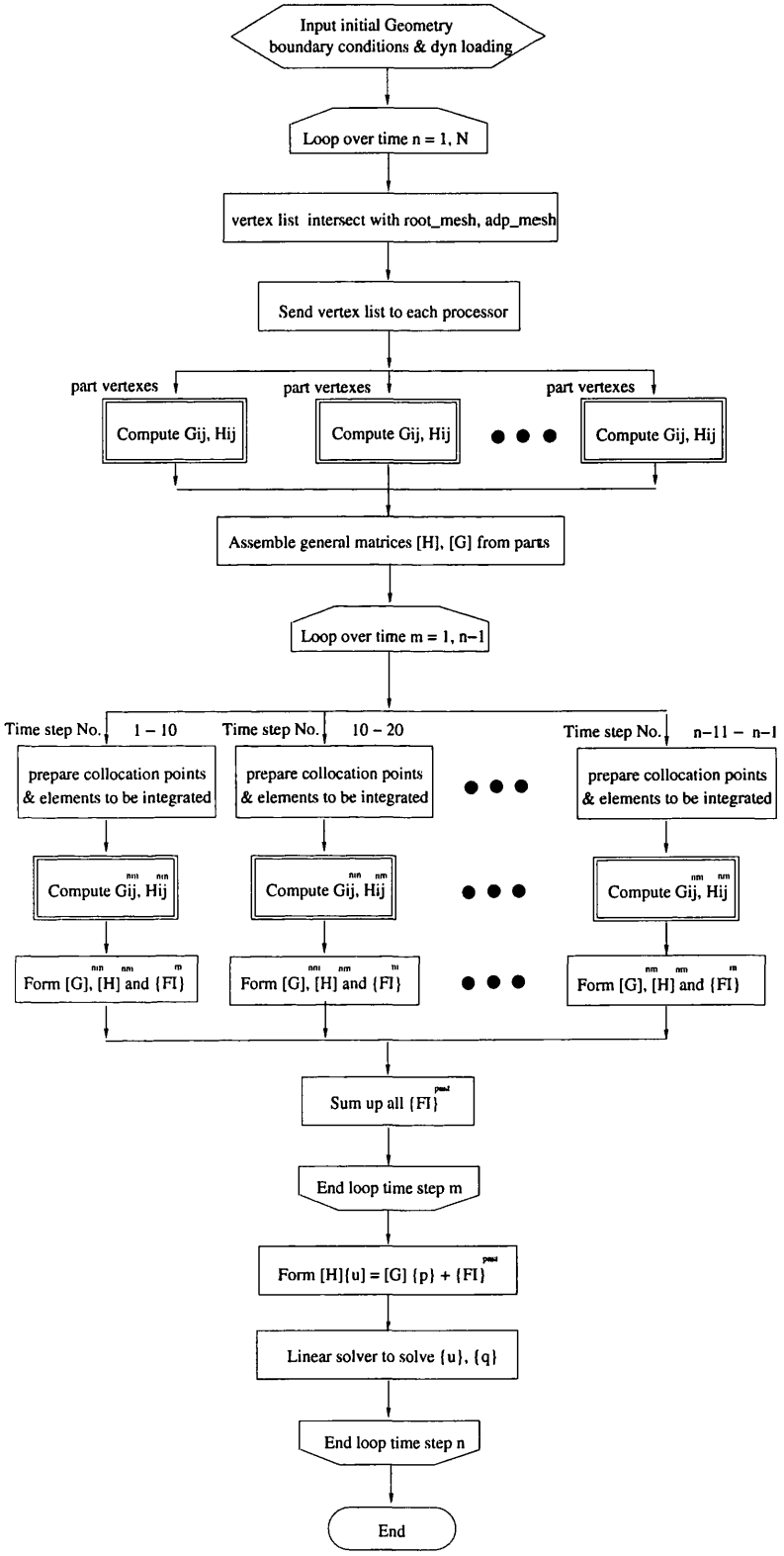


Figure 7.4: Parallel process of dynamic BEM

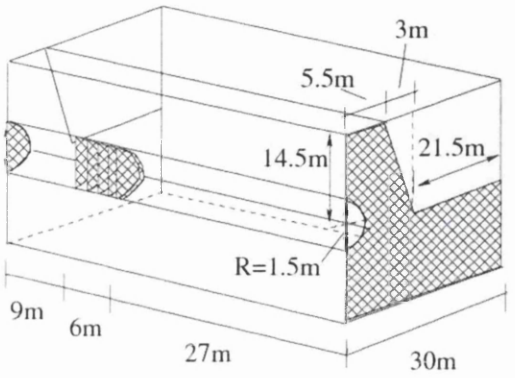


Figure 7.5: Geometry of the riverbank-tunnel system

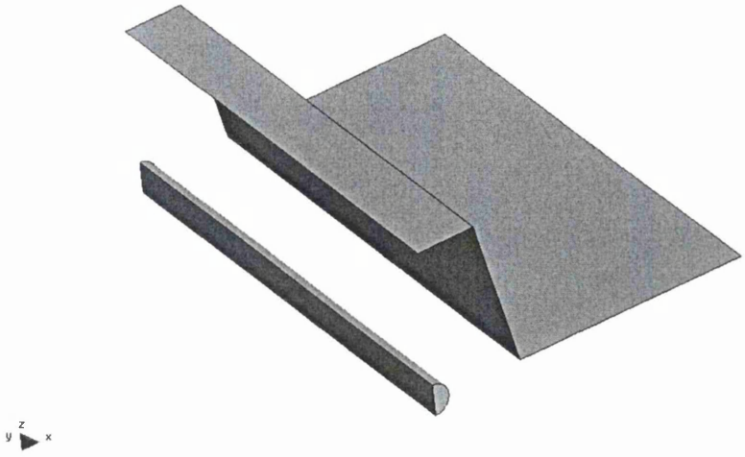


Figure 7.6: Gid model of the riverbank-tunnel system

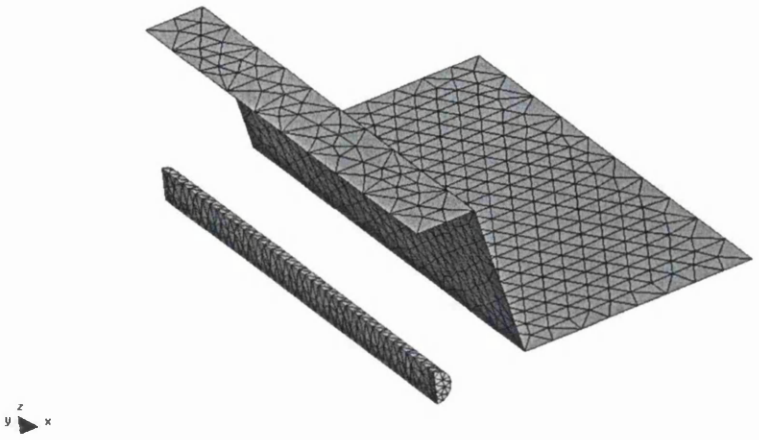


Figure 7.7: Mesh of the riverbank-tunnel system

	CPU time (s)	Speed-up	Efficiency
1 CPU	43,542	1.00	1.00
2 CPUs	31,453	1.38	0.69
4 CPUs	19,765	2.20	0.55
8 CPUs	10,332	4.21	0.52

Table 7.1: Performance of Parallel BEM algorithm

the speed-up per processor:

$$Efficiency = \frac{speedup}{number\ of\ processors\ used} \tag{7.8}$$

The efficiency of the parallel algorithm is shown in Table 7.1, for up to eight processors. Compared with one processor, when all 8 processors are employed, the speed-up is approximately four, yielding an efficiency of more than 50%. The trend suggests that this type of problem is very well suited for parallelization and that many more processors could be employed to solve a large problem effectively.

7.5 Summary

An implementation of parallel BEM on a Linux cluster has been developed to reduce computational time in dynamic BEM. The results suggest that efficiency of the dynamic BEM solver can be further improved if the necessary hardware is available.

Chapter 8

Conclusions

8.1 Conclusions

From the literature review in Chapter 2, several problems are apparent in the conventional dynamic BEM. Firstly, the method is expensive since influence matrices are computed at each time step and BEM solutions at every former time step have to be stored. Secondly, if large time steps are used, inaccuracies arise in BEM solution; but if small time steps are used, computational costs become impractical. Thirdly, the dimensionless space-time ratio in the conventional BEM must be limited to a narrow range (roughly 0.3 - 1.5) to produce a stable and accurate solution.

The basic strategy to attack these problems is to introduce adaptive schemes and mesh refinement to the dynamic BEM. Instead of using uniform meshes and uniform time steps, error indicators are employed to locate high-gradient areas; then mesh refinement in space-time is used to improve the resolution in those areas only. Another strategy is to introduce the space-time concept to track moving wavefronts. In wave problems, wavefronts move in space-time, and high gradients appear both in space and in time. It is thus inadequate to refine the mesh in space only because there are high gradients in time as well. Hence, besides a locally mesh refinement scheme employed in space, local time stepping is also used to improve the accuracy and efficiency of the algorithm.

8.1.1 2D scalar wave problems

Scalar wave problems in 2D are solved using the strategy mentioned above. A new formula in 2D scalar wave BEM has been derived by the author to make it possible to use adaptive schemes in space-time. Then, gradient-based and resolution-based error indicators are employed to locate those moving high-gradient areas. A h- adaptive mesh refinement scheme is used to refine the area to achieve higher accuracy. Using a benchmark of a 2D bar under impact load, the adaptive BEM solver is $30\% \sim 40\%$ times faster than the conventional BEM solver. It is also more stable than the conventional BEM.

8.1.2 3D scalar wave problems

A significant difference between 2D and 3D problems is that in 3D BEM integration is necessary over only part of the boundary, and the part which has to be integrated changes in each time step. This requires a totally different BEM solution for the time and space integrals. Elements in space are indexed, like books in a library, to accelerate the spatial search to determine which part of the boundary mesh should be integrated in different time steps. This greatly improves efficiency of the dynamic BEM solver particularly if the total number of elements is large.

Difficulties arise when dealing with non-uniform meshes. Conventionally, we either lose accuracy by using large time steps, or lose stability by using small ones. Adaptive BEM solvers are the right solution for the wave problem with non-uniform meshes. By starting with uniform time steps, local time stepping is triggered which offer different time steps to elements with different sizes controlled by error estimates.

It appears that the impulse wave propagation problem (labeled as an unsolved problem by Prof. Zienkiewicz) may be attacked by employing adaptive BEM wave solvers. Using the benchmark of a 3D bar under impact loads, the adaptive BEM solver is about $20\% \sim 45\%$ times faster than the conventional BEM solver, and is also more stable.

8.1.3 3D elastodynamics

The same adaptive approach is applied to the BEM integral equation for 3D elastodynamic problems as for the scalar problem. Further work has been done to compute the space integrals and the time integrals for vector fields, and how to compute the boundary stresses. A similar improvement in efficiency is also obtained here.

For conventional BEM solvers, the proper space/time ratio β in 3D elastodynamics is more limited than that which applied for scalar wave problems, because there are now two different wave speeds. The adaptive BEM solvers bypass this problem in 3D elastodynamics by using local time stepping, which in theory can even assign different time steps for different wave speeds. The overall effects of adaptivity is to increase stability.

8.1.4 Parallel dynamic BEM

Many computational procedures in dynamic BEM are independent from each other and thus inherently parallelizable. Examples are computing influence matrices at different time steps; computing entries in influence matrices and the function evaluations in the numerical integrals. All these processes can be easily paralleled by partitioning the overall process into separate tasks; allocating tasks to different processors and synchronizing the tasks to obtain the final results.

We apply the parallel BEM to solve the problems of a 3D underground explosion. Compared with the non-parallel code, using a 8-processor Linux cluster, the *"speed-up"* is factor of four. The trend suggests that further *"speed-up"* is possible if more processors are employed.

8.1.5 Final remarks

For the scalar or elastodynamic 2D or 3D wave propagation problems in open or closed fields, conventional BEM dynamic solvers are usually inefficient and unstable for impulsive loading or complex geometries. Through deriving new BEM formulas for adaptive schemes and solving several numerical examples, we show that adaptive dynamic BEM solver offers more efficient, more accurate and more stable solution.

8.2 Suggestions for further research

8.2.1 Time-domain elastoplastic BEM

In some situations it is desirable to use elastoplastic material models to allow more realistic simulations. A time-domain BEM solver is necessary for this purpose. It is not only capable of solving linear elastodynamics, but also paves the way for solving the nonlinear problems too. In the basic theory of elastoplasticity, it is assumed that the strain is infinitesimal; the strain tensor can be decomposed into an elastic part and a plastic part , and the elastic strain tensor follows the linear constitutive relation

$$\varepsilon = \varepsilon^E + \varepsilon^P \qquad where \qquad \sigma = C : \varepsilon^E \tag{8.1}$$

or

$$\sigma = C : \left(\varepsilon - \varepsilon^P\right) \tag{8.2}$$

Hence the strategy of solving nonlinear problems using BEM is to set up boundary integral equations which govern elastic problems with distributions of *a priori* unknown initial strains or stresses (see Fig. 8.1):

$$\begin{aligned} c^i_{lk} u^i_k \left(\boldsymbol{x}^i, t\right) &= \int_S u^*_{lk} \left(\boldsymbol{x}, t-\tau; \boldsymbol{x}^i\right) * p_k \left(\boldsymbol{x}, \tau\right) dS \left(\boldsymbol{x}, \tau\right) \\ &- \int_S p^*_{lk} \left(\boldsymbol{x}, t-\tau; \boldsymbol{x}^i\right) * u_k(\boldsymbol{x}, \tau) dS(\boldsymbol{x}, \tau) + \\ &+ \int_{V^P} \varepsilon^*_{ljk} \left(\boldsymbol{x}, t-\tau; \boldsymbol{x}^i\right) * \sigma^P_{jk}(\boldsymbol{x}, \tau) dS(\boldsymbol{x}, \tau) \end{aligned} \tag{8.3}$$

These equations are discretized as before, but include a new term of domain integrals containing σ^P. The boundary integral equations and their discrete counterparts have to be solved by iteration because the equations are no longer linear. Since the plastic region, where the plastic deformation occurs, is not known *a priori*, it may be adaptively updated at each step of the iteration. The adaptive schemes proposed in this thesis could play this role.

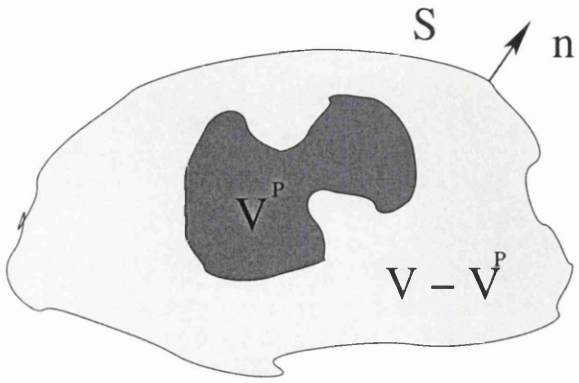


Figure 8.1: Domain consists of both plastic region $V^P$and elastic region $V - V^P$

8.2.2 Wavelet-based adaptive schemes for time-domain BEM

During this research, the author found that mesh-based adaptive schemes have drawbacks. First of all, the construction of adaptive meshes is a laborious task. Elements with large errors as well as their neighbouring elements have to be divided into smaller elements, and relevant geometric information such as their neighbouring elements have to be updated after each refinement. The computer implementation is highly complex because old elements have to be physically removed from the list of elements, and new elements and their vertexes have to be inserted into the element list and the vertex list separately. Secondly, the refinement ratio between new elements and old ones should take a reasonable value (less than ten, say). If the ratio is too large, the resulting linear system equations will become ill-conditioned, and yield unreliable solutions.

Some non-mesh based adaptive schemes with new shape functions can be used to overcome these difficulties. Just like polynomials, wavelet functions are a series of complete, orthonormal multi-scale functions which can be used to approximate any arbitrary function. Because of their orthonormal property (i.e., any function is orthogonal to other functions in the functional space), adaptivity can be achieved by applying another set of functions without removing the old ones. Also, since the function itself is multi-scale, the coefficients will be approximately of the same scale and the problem of ill-conditioning will be avoided.(Dahmen, Kurdila and Oswald [17])

The starting point of wavelet-based adaptive schemes is the same as the conven-

tional time-domain BEM solver. The boundary integral equations are discretized on the boundary and shape functions are used to transform the arbitrary geometry into regular geometric shapes, such as unit squares. Unlike the conventional isoparametric method (which applies the same shape functions to physical quantities and geometries), in wavelet-based adaptive schemes, those physical quantities are approximated by using wavelet-like shape functions. If element solutions are found to have large errors, wavelet-like shape functions (at a smaller scale) can be added to offer a better resolution in these high-gradient areas. The process is repeated until the prescribed error tolerance is satisfied. This promises to be a fruitful area for future research.

<END>

Bibliography

[1] S. Adjerid and J. E. Flaherty. A moving fem with error estimation and refinenment for 1-d time dependent pdes. *SIAM Journal on Numerical Analysis*, 23(4):778–796, 1986.

[2] S Ahmad and P K. Banerjee. Time domain transient elastodynamic analysis of 3-d solids by bem. *Int J Num Meth Engng*, 26(8):1709–1728, 1988.

[3] Sollero P. Albuquerquea, E. L. Dual reciprocity boundary element method in laplace domain applied to anisotropic dynamic crack problems. *Computers & Structures*, 81(17):1703–1713, 2003.

[4] R. P. Banaugh and W. Goldsmith. Diffraction of steady acoustic waves by surfaces of arbitrary shape. *Journal of the Acoustical Society of America*, 35(10):1590–1601, 1963.

[5] Marc Bernacki, Loula Fezoui, Stéphane Lanteri, and Serge Piperno. Parallel discontinuous galerkin unstructured mesh solvers for the calculation of three-dimensional wave propagation problems. *Applied Mathematical Modelling*, 30(8):744–763, 2006.

[6] G. Beylkin. *On wavelet-based algorithms for solving differential equations, Wavelets: mathematics and applications, Stud. Adv. Math.*

[7] Crouch SL. Birgisson B. Elastodynamic boundary element method piecewise homogeneous media. *International Journal for Numerical Methods in Engineering*, 42:1045–1069, 1998.

[8] C. A. Brebbia and J. Dominguez. *Boundary Elements: An Introductory Course.* Mcgraw-Hill College, 1989.

[9] Laura A. Brooks and Colin H. Hansen Rick C. Morgans. Learning acoustics through the boundary element method: an inexpensive graphical interface and associated tutorials. *Acoustics Australia*, 33(3):89–95, 2005.

[10] A. Burbeau and P. Sagaut. Simulation of a viscous compressible flow past a circular cylinder with high order discontinuous galerkin methods. *Computers & Fluids*, 31(8):867–889, 2002.

[11] M. F. Calitz, A. G. DuToit, and W. Drijfhout. Analysis of high-perveance electron guns. *Computers in Physics*, 4(3):280–284, 1990.

[12] J. A. M. Carrer and W. J. Mansur. Stress and velocity in 2d transient elastodynamic analysis by the boundary element method. *Engineering Analysis with Boundary Elements*, 23(3):233–245, 1999.

[13] Jack Chessa and Ted Belytschko. An enriched finite element method and level sets for axisymmetric two-phase flow with surface tension. *International Journal for Numerical Methods in Engineering*, 58(13):2041–2064, 2003.

[14] Chyou C. Chien, Yu H. Chen, and Ching C. Chuang. Dual reciprocity bem analysis of 2d transient elastodynamic problems by time-discontinuous galerkin fem. *Engineering Analysis with Boundary Elements*, 27(6):611–624, 2003.

[15] Kosloff DD Minster JB Cole, DM. A numerical boundary integral method for elastodynamics. i. *Bulletin of the Seismological Society of America*, 68:1331–1357, 1978.

[16] T A Cruse and F J Rizzo. A direct formulation and numerical solution of the general transient elastodynamic problem. i (transient problem solution in linear elastodynamics by integral equation in laplace transform space). *Journal Of Mathematical Analysis And Applications*, 22:244–259, 1968.

[17] Wolfgang Dahmen, Andrew Kurdila, and Peter Oswald. *Multiscale Wavelet Methods for Partial Differential Equations (Wavelet Analysis and Its Applications).* Academic Press, 1997.

[18] A. J. Davies. Fine-grained parallel boundary elements. *Engineering Analysis with Boundary Elements*, 19(1):13–16, 1997.

[19] Duncan D. B. Davies P. Numerical stability of collocation schemes for time domain boundary integral equations. *Computational Electromagnetics*, 28:51–66, 2003.

[20] J. Dominguez. *Boundary elements in dynamics.* Computational Mechanics Publications, Southampton, 1993.

[21] S. Mallat E. Bacry and G. Papanicolaou. A wavelet based space-time adaptive numerical method for partial differential equation. *Mathematical Modelling and Numerical Analysis*, 26(7):793–796, 1992.

[22] E.Bécache, P.Joly, and J.Rodrý. Space - time mesh refinement for elastodynamics. numerical results. *Computer Methods in Applied Mechanics and Engineering*, 194(2-5):355–366, 2004.

[23] E.Kita and N.Kamiya. An overview, error estimation and adaptive mesh refinement in bem. *Engineering Analysis with Boundary Elements*, 25(7):479–495, 2001.

[24] E.Oñate, F.Perazzo1, and J.Miquel. A finite point method for elasticity problems. *Computers & Structures*, 79(22-25):2151–2163, 2001.

[25] Charbel Farhat, Isaac Hararib, and Leopoldo P.Francac. The discontinuous enrichment method. *Computer Methods in Applied Mechanics and Engineering*, 190(48):6455–6479, 2001.

[26] A. Frangi and G. Novati. On the numerical stability of time-domain elastodynamic analyses by bem. *Computer Methods in Applied Mechanics and Engineering*, 173(3-4):403–417, 1999.

[27] X.W. Gao and T. G. Davies. *Boundary Element Programming in Mechanics.* Cambridge University Press, New York, 2002.

[28] Prof.Lothar Gaul. Boundary element methods. *Internet, http://www.bem.uni-stuttgart.de/home.htm*, 2004.

[29] Personal pre Gid and http://gid.cimne.upc.es/ post processor. http://gid.cimne.upc.es/.

[30] M. A. Golberg and H. Bowman. Superconvergence and the use of the residual as an error estimator in the bem. ii: Collocation, numerical integration and error indicators. *Engineering Analysis with Boundary Elements*, 25(7):511–521, 2001.

[31] Patricia González, Tomás F.Pena, and José C.Cabaleiro. Parallel sparse approximate preconditioners applied to the solution of bem systems. *Engineering Analysis with Boundary Elements*, 28(9):1061–1068, 2004.

[32] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.

[33] Nowak Z.P. Hackbusch, W. On the fast matrix multiplication in the boundary element method by panel clustering. *Numer. Math.*, 54:463–491, 1989.

[34] H.Huang and F.Costanzo. On the use of space-time finite elements in the solution of elasto-dynamic problems with strain discontinuities. *Computer Methods in Applied Mechanics and Engineering*, 191(46):5315–5343, 2002.

[35] G.C. Hsiao and W.L. Wendland. *Boundary element methods: Foundation and error analysis, Encyclopedia of Computational Mechanics.*

[36] Philip M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Trans. Graph.*, 15(3):179–210, 1996.

[37] Thomas J. R. Hughes and Gregory M. Hulbert. Space-time finite element methods for elastodynamics: Formulations and error estimates. *Computer Methods in Applied Mechanics and Engineering*, 66(3):339–363, 1988.

[38] A. S. M. Israil and P. K. Banerjee. Two-dimensional transient wave-propagation problems by time-domain bem. *International Journal of Solids and Structures*, 26(8):851–864, 1990.

[39] MA Jaswon. Integral equation methods in potential theory 1. *Proc. Roy. Soc. Series A*, 275:23–32, 1963.

[40] MA Jaswon and AR Ponter. An integral equation solution of the torsion problem. *Proc. Roy. Soc. Series A*, 273:237–246, 1963.

[41] I. B. J.M.Melenk. The partition of unity finite element method: Basic theory and applications. *Computer Methods in Applied Mechanics and Engineering*, 139(1-4):283–314, 1996.

[42] C. Johnson. Discontinuous galerkin finite element methods for second order hyperbolic problems. *Computer Methods in Applied Mechanics and Engineering*, 107:117–129, 1993.

[43] D. L. Karabalis. A simplified 3-d time domain bem for dynamic soil-structure interaction problems. *Engineering Analysis with Boundary Elements*, 8(3):139–145, 1991.

[44] H. Kawaguchi. Time-domain analysis of electromagnetic wave fields by boundary integral equation method. *Engineering Analysis with Boundary Elements*, 27(4):291–304, 2003.

[45] E. Kita, K. Higuchi, and N. Kamiya. Application of r- and hr-adaptive bem to two-dimensional elastic problem. *Engineering Analysis with Boundary Elements*, 24(4):317–324, 2000.

[46] E. Kita and N Kamiya. An overview, error estimation and adaptive mesh refinement in BEM. *Engineering Analysis with Boundary Elements*, 25(7):479–495, 2001.

[47] M. Kreienmeyer and E. Stein. Efficient parallel solvers for boundary element equations using data decomposition. *Engineering Analysis with Boundary Elements*, 19(1):33–39, 1997.

[48] http://www.netlib.org/lapack/ Lapack Development team. http://www.netlib.org/lapack/, 1974.

[49] Robert L.Taylor, O.C.Zienkiewicz, and E.Oñate. A hierarchical finite element method based on the partition of unity. *Computer Methods in Applied Mechanics and Engineering*, 152(1-2):63–84, 1998.

[50] J. C. F. Telles M. T. F. Cunha and A. L. G. A. Coutinho. A portable parallel implementation of a boundary element elastostatic code for shared and distributed memory systems. *Advances in Engineering Software*, 35(7):453–460, 2004.

[51] G. Manolis and D. Beskos. *Boundary Element Methods in Elastodynamics.* Unwin Hyman, London, 1988.

[52] Beskos DE. Manolis GD. Dynamic stress concentration studies by boundary integrals and laplace transform. *International Journal for Numerical Methods in Engineering*, 17:573–599, 1981.

[53] W. J. Mansur and J. A. M. Carrer. Higher order traction time approximation in time domain scalar wave equation analysis. Proceedings of the 1997 19th International Conference on Boundary Element Methods, BEM, Sep 9-12 1997, Rome, Italy:107–116, 1997.

[54] W.J. Mansur and C. A. Brebbia. *Topics in Boundary Element Research*, volume 2: Time-dependent and Vibration Problems. Springer-Verlag, Berlin, 1985.

[55] M.Marrero and J.Dominguez. Numerical behavior of time domain bem for three-dimensional transient elastodynamic problems. *Engineering Analysis with Boundary Elements*, 27(1):39–48, 2005.

[56] K. H. Muci-Küchler and J. C. Miranda-Valenzuela. A new error indicator based on stresses for three-dimensional elasticity. *Engineering Analysis with Boundary Elements*, 25(7):535–556, 2001.

[57] Nishimura N. Fast multipole accelerated boundary integral equation methods. *Applied Mechanics Reviews*, 55:299–324, 2002.

[58] D. Nardini and C. A. Brebbia. *A New Approach for Free Vibration Analysis Using Boundary Elements,Boundary Element Methods in Engineering.* Springer, Berlin, 1982.

[59] N.Heuer, M.E.Mellado, and E.P.Stephan. hp-adaptive two-level methods for boundary integral equations on curves. *Computing*, 67(4):305–334, 2005.

[60] Y Niwa, S Kobayashi, and N. Azuma. An analysis of transient stresses produced around cavities of arbitrary shape during passage of travelling waves. *Memoirs of the Faculty of Engineering*, 36(2):28–46, 1975.

[61] Zienkiewicz OC and Zhu JZ. The superconvergent patch recovery and a posteriori error estimates.part 1: The recovery technique. *Int. J. Numer. Methods Engrg.*, 33(4):1331–1364, 1992.

[62] P.A. Raviart P. x. *On a finite element method for solving the neutron transport equation,Mathematical Aspects of Finite Elements in Partial Differential Equations.*

[63] G. H. Paulino, F. Shi, S. Mukherjee, and P. Ramesh. Nodal sensitivities as error estimates in computational mechanics. *Acta Mechanica*, 121(1):191–219, 2006.

[64] A Peirce and E Siebrits. Stability analysis and design of time-stepping schemes for general elastodynamic boundary element models. *International Journal for Numerical Methods in Engineering*, 40(2):319 – 342, 1997.

[65] E. Perrey-Debain, J. Trevelyan, and P. Bettess. Wave boundary elements: a theoretical overview presenting applications in scattering of short waves. *Engineering Analysis with Boundary Elements*, 28(2):131–141, 2004.

[66] Christophe Peyret Philippe Delorme, Pierre Mazet and Yoan Ventribout. Computational aeroacoustics applications based on a discontinuous galerkin method. *Comptes Rendus Mecanique*, 333(9):676–682, 2005.

[67] S. Piperno. Dgtd methods using modal basis functions and symplectic local time stepping: application to wave propagation problems. *European Journal of Computational Mechanics*, 15(6):643–670, 2006.

[68] C. Pozrikidis. *A Practical Guide to Boundary Element Methods with the Software Library BEMLIB.* CRC, 2002.

[69] M. Rivara and N. Hitschfeld. Leppdelaunay algorithm: a robust tool for producing size-optimal quality triangulations, 1999.

[70] D. C. Rizos and D. L. Karabalis. Advanced direct time domain bem formulation for general 3-d elastodynamic problems. *Computational Mechanics*, 15(3):249–269, 1994.

[71] D.C. Rizos and D.L. Karabalis. Advanced direct time domain bem formulation for general 3-D elastodynamic problems. *Computational Mechanics*, 15(3):249–269, 1994.

[72] FJ Rizzo. An integral equation approach to boundary value problems of classical elastostatics. *Q. Appl. Math.*, 25:83–95, 1967.

[73] Marcus Rüter and Erwin Stein. *Finite Element Method for Elasticity with Error-controlled Approximation and Model Adaptivity.* John Wiley & Sons, New York, 2004.

[74] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1990.

[75] http://www.netlib.org/scalapack/ Scalpack Development team. http://www.netlib.org/scalapack/, 1974.

[76] Dominik Schőtzau and Christoph Schwab. hp-discontinuous galerkin time-stepping for parabolic problems. *Comptes Rendus de l Acadẽmie des Sciences - Series I - Mathematics*, 333(12):1121–1126, 2001.

[77] J. Simkin and C. W. Trowbridge. Magnetostatic fields computed using an integral equation derived from green's theorem. *Compumag Conference on the Computation of Magnetic Fields*, 1976.

[78] J.J. Sudirham, J.J.W. van der Vegt, and R.M.J. van Damme. Space - time discontinuous galerkin method for advection diffusion problems on time-dependent domains. *Applied Numerical Mathematics*, 56(12):1491–1518, 2006.

[79] GT Symm. Integral equation methods in potential theory 2. *Proc. Roy. Soc. Series A*, 275:33–46, 1963.

[80] N. Nishimura T. Takahashi and S. Kobayashi. A fast biem for three-dimensional elastodynamics in time domain. *Engineering Analysis with Boundary Elements*, 28(2):165–180, 2004.

[81] G. R. C. Tai and R. P. Shaw. Diffraction of steady acoustic waves by surfaces of arbitrary shape. *Journal of the Acoustical Society of America*, 56(3):796–804, 1974.

[82] T.Ha-Duong, B.Ludwig1, and I.Terrasse2. A galerkin bem for transient acoustic scattering by an absorbing obstacle. *International Journal for Numerical Methods in Engineering*, 57:1845–1882, 2005.

[83] Timoshenko and Goodier. *Theory of Elasticity, 3rd ed.* New York: McGraw-Hill, 1970.

[84] S. V. Tsinopoulos, S. E. Kattis, D. Polyzos, and D. E. Beskos. An advanced boundary element method for axisymmetric elastodynamic analysis. *Computer Methods in Applied Mechanics and Engineering*, 175(1-2):53–70, 1999.

[85] S. P. Walker and M. J Bluck. The stability of integral equation time-domain scattering computations for three-dimensional scattering; similarities and differences between electrodynamic and elastodynamic computations. *International Journal of Numerical Modelling Electronic Networks Devices and Fields*, 15(5-6):459–474, 2002.

[86] C. C. Wang and H. C. Wang. Two-dimensional elastodynamic transient analysis by QL time-domain BEM formulation. *International Journal for Numerical Methods in Engineering*, 39(6):951–985, 1996.

[87] Nils E. Wiberg, Lingfu Zeng, and Xiangdong Li. Error estimation and adaptivity in elastodynamics. *Computer Methods in Applied Mechanics and Engineering*, 101(1-3):369–395, 1992.

[88] T. W. Wu. *Boundary Element Acoustics: Fundamentals and Computer Codes.* Wit Pr/Computational Mechanics, 2001.

[89] G. Y. Yu and W.J. Mansur. Linear θ method applied to 2D time-domain BEM. *Communications in Numerical Methods in Engineering*, 14(12):1171–1179, 1998.

[90] Guoyou Yu, W. J. Mansur, J. A. M. Carrer, and S. T. Lie. A more stable scheme for bem/fem coupling applied to two-dimensional elastodynamics. *Computers & Structures*, 79(8):811–823, 2001.

[91] Zhiye Zhao and Xin Wang. Error estimation and h adaptive boundary elements. *Engineering Analysis with Boundary Elements*, 23(10):793–803, 1999.

[92] O.C. Zienkiewicq. Achievements and some unsolved problems of the finite element method. *International journal for numerical methods in engineering*, 47:9–28, 2000.

[93] Afra Zomorodian, Herbert E' delsbrunner, and Mark De Berg. Fast Software for Box Intersections. *International Journal of Computational Geometry and Applications*, 12:143–173, 2002.

[94] Z.Yue and D.H.Robbins Jr. Adaptive superposition of finite element meshes in elastodynamic problems. *International Journal for Numerical Methods in Engineering*, Published Online: 13 Apr 2005, Early View (Articles online in advance of print), 2005.