Soil-Structure Interaction Analysis Using High-Performance Parallel Computation

by

Georgios Petropoulos

M.S. (University of California, Berkeley) 2002

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Engineering-Civil and Environmental Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Gregory L. Fenves, Chair
Professor Bozidar Stojadinovic
Professor James W. Demmel

Fall 2008

UMI Number: 3353077

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI®

UMI Microform 3353077

Copyright 2009 by ProQuest LLC.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest LLC
789 E. Eisenhower Parkway
PO Box 1346
Ann Arbor, MI 48106-1346

Soil-Structure Interaction Analysis Using High-Performance Parallel Computation

Copyright © 2008

by

Georgios Petropoulos

Abstract

Soil-Structure Interaction Analysis Using High-Performance Parallel Computation

by

Georgios Petropoulos

Doctor of Philosophy in Engineering-Civil and Environmental Engineering

University of California, Berkeley

Professor Gregory L. Fenves, Chair

The primary objective of this dissertation is to design and develop high-performance algorithms and software for the simulation of soil-foundation-structure interaction on large soil domains using finite elements and mixed time integration on parallel computers. Second, the dissertation presents a preliminary study, conducted with numerical simulations, of the response of simplified structural models to near-fault, pulse-type ground motion for varying soil conditions.

The development of a scalable and efficient finite element software, is based on the external loading computation, the element state determination, the mixed time integration scheme and its implementation, and finally the communication scheme and the solution phase. The object-oriented paradigm is used for the software design because it allows for usage of existing software components for the remaining parts of the software that are modified for performance and scalability reasons.

The external loading computation uses a novel method of effective seismic input, the Domain Reduction Method, which requires processing of large data sets that represent

the wave field used to generate the input forces. Specialized classes are developed to implement this loading pattern.

Explicit time integration with diagonal mass matrices is selected for regions dominated by wave propagation because it does not require factorization of the dynamic stiffness matrix, it has a minimal memory footprint, and it is highly scalable. To analyze structures implicit time integration is needed, and therefore the coupled problem is solved using a mixed time integration algorithm that is extended to handle nonlinear systems in the implicit partition by combining it with Newton-Raphson iterations. The software design for this algorithm allows for various types of communication between the implicit and explicit partitions via subclassing and use of virtual methods, for example within the same address space, within the same communicator or between subcommunicators of a communicator.

The communication scheme used by the explicit time integration solver is designed to be tunable depending on the problem size, number of processors, hardware platform and message passing library. An efficient numbering scheme is presented for general purpose finite element degree of freedom numbering which is almost embarassingly parallel. Two methods for setting up the communication graph are presented, one based on the all-to-all multicast and one based on the one-to-all multicast.

The second part of the dissertation consists of computational simulations of the soil-foundation-structure-interaction response of simple structural models on large soil domains. Initially the response of a large, near-fault soil domain to low-frequency pulse-type input is studied, and then compared to broadband input. The soil simulation for low-frequency input is repeated for a softer soil profile and increased spatial variability over the surface of the soil domain is observed.

The low-frequency input is used to shake the same soil volume with single simplified structural models for two and four second vibration period systems. The results indicate that the simplified soil-foundation-structure interaction closed-form solu-

tions tend to overestimate the effects of this interaction as far as period lengthening and damping. For the four second period systems reduction of the effective damping is observed which is due to rocking. Significant variability on the permanent offsets of yielding structures which have the same fundamental elastic period of vibration is observed. The interaction in most cases reduces the structural deformations. Finally, two building interaction analyses are conducted to examine the ensemble effects of interaction between buildings and to examine the scalability of the mixed integration method.

Professor Gregory L. Fenves
Dissertation Committee Chair

To my parents and the memory of my grandmother Antigoni

Στους γονείς μου Νίκο και Φρόσω και στη μνήμη της γιαγιάς μου Αντιγόνης

Contents

List of Figures

List of Tables

Listings

Acknowledgements

I would like to thank my advisor Dean Gregory L. Fenves for guiding me through my studies and for supporting my research work. His expertise, vision and professionalism made my study and research years a rewarding and fulfilling journey to knowledge and scientific endeavor.

I would also like to thank the members of my dissertation committee: Professor Bozidar Stojadinovic and Professor James W. Demmel, for their time and effort to provide me with comments and advice.

I would like to extend my thanks to Professor Jacobo Bielak for providing me with the input wavefields used in my simulations, and for sharing his expertise on wave propagation and the Domain Reduction Method.

Thanks to my advisor's vision i developed a strong interest in Computer Science that led to a Master's degree with a project under the supervision of Professor James W. Demmel. I would like to thank Professor James W. Demmel for offering me the opportunity to learn from his expertise in Scientific Computing, Computer Science and Applied Mathematics.

I would also like to thank Professor Katherine Yelick for reading my Master's Thesis very quickly.

I would like to thank the beboppers and especially Mark Hoemmen.

I would like to thank my friends: Nick, Yalin, Christian, Alidad, Miltos, Frank, Jaesung, Zoe, Ellie, Matthew, Chin-Long, Shamim, Mohammad and Gabriel.

I would also like to thank my aunt Varvara for her love, support and encouragement.

I would like to thank my brother Alexandros whom i still have to visit in Sweden.

Certainly most importantly, i would like to thank my parents Nikos and Frosso, whose unconditional love has been the greatest support i could ever hope for. Their patience,

encouragement, and believing in me during all these years made this possible. I hope i can become nearly as good of a parent as they succeeded to be.

Chapter 1

Introduction

Earthquakes are a constant threat in large areas of the United States and around the world. Assessing the risks and mitigating the hazards of earthquakes is one of the primary missions of the earthquake engineering community. Estimating the ground motion to which structures will be exposed during their lifetimes and predicting their responses to this motion is essential in designing and retrofitting earthquake-resistant buildings and infrastructure. Determining performance of structures, including damage and loss, is also of great use for emergency planning and management purposes. In the context of these societal needs, the research project Seismic Performance Of Urban Regions (SPUR) has been conceived. The goal of SPUR is to create a powerful new computational modeling and simulation system for understanding the impact of an earthquake on a region, for use by public policy makers and earthquake engineering researchers. SPUR aims to achieve this by simulating the performance of the infrastructure of an urban region subject to an earthquake scenario. The ultimate goal is to use simulation coupled with visualization to examine the amount, distribution and type of damage to buildings, bridges, and lifelines caused by a realistic earthquake scenario in an actual urban site, and provide decision-makers and stakeholders a better understanding of this complex phenomenon.

SPUR's approach can be divided into three major areas:

1. Earthquake ground motion for a prescribed earthquake source and geological structure: simulation of ground motion using three dimensional finite element and wave propagation analysis from the fault to the area of interest.

2. Structural modeling of the dynamic response of realistic-based building models for the inventory within a region of interest: A smaller domain with appropriate boundary conditions and a set of structural models is analyzed using three-dimensional finite elements to account for site effects and soil-foundation-structure interaction (SFSI) to the dynamic response of a given structure.

3. Visualization tools for the soil and structures as part of a three-dimensional domain.

1.1 Problem statement

The objective of this research is to develop an efficient parallel computational method for the regional SFSI problem using representative simplified structural models, three-dimensional finite element models of a region of interest, realistic earthquake scenarios and investigate factors increasing the complexity of the structural response such as near-fault ground motions and local site-effects. This is a sub-region of the system that includes the fault, the propagation path of the excitation up to the region of interest, and the region of interest along with the buildings and foundations. The scientific goals are to model a three-dimensional subdomain with appropriate boundary conditions, account for the proper wave propagation and energy transfer from and to the half-space that represents the soil, and using structural models simulate, study and provide a better understanding of the dynamic response of structures accounting for local site effects, complex SFSI and the effects of near fault ground motion.

To enable the simulation of a sub-region of soil, foundations and structures at the desired scale, high performance and scalable computational tools are necessary. The advances in the field of computer science and specifically hardware architecture, networks,

programming languages and operating systems have enabled the widespread of parallel or distributed memory computing, which is the computational platform that enables large scale scientific computations. Massive computational power is available today to researchers enabling them to analyze larger problems than ever at unparalleled computational speeds. At the same time the requirements within this computational environment are for high performance, scalability and efficiency which allow the better utilization of the computing resources.

The following components of the problem are identified and addressed in this research:

- Modeling: layered soil models that accurately describe the site response for a subregion; provide effective seismic input for the soil subdomain, and enable simulation for different scenarios; appropriate boundary conditions that account for the fact that the soil is a semi-infinite layered halfspace; structural models of the superstructure and foundation that coupled with the soil will enable the study of the SFSI problem.

- Computational efficiency: since a large problem with many degrees of freedom (in the order of hundreds of millions) is to be solved, efficiency and speed of parallel solution are essential. Algorithmic and data structures issues are to be addressed for efficient computational strategies. Explicit and mixed explicit-implicit time integration of domains must be addressed because of the different physics for wave propagation in the soil and vibration in the buildings. Scalability of the application is required to be able to solve very large problems with full utilization of massive numbers of processors.

- Software implementation: utilization of the object-oriented programming paradigm, the message passing interface and customized, platform-dependent tuning, to develop a custom application of high-performance.

- Study of various scenarios: varying of soil parameters, structural properties, and input intensities to study SFSI in the regional scale. The influence of softening the

soil profile is examined, the characteristics of low frequency versus broadband input, and the seismic performance of structural models with varying structural properties are studied.

1.2 Factors influencing the dynamic response of a building

1.2.1 Ground motion

The factors that affect the strong motion recording are : (1) basin effects, (2) local site effects and (3) surface topography.

The term basin effects refers to the potential amplification and time elongation of ground motions that can occur when waves become trapped in deep sedimentary basins. A basin consists of alluvial deposits and sedimentary rocks that are geologically younger and have lower seismic wave velocities than the underlying rocks upon which they have been deposited. Waves that become trapped in deep sedimentary basins can produce up to 50% stronger amplitudes at intermediate and low frequencies ($f < 1$ Hz) and elongate their duration up to about two times (Graves et al., 1998), compared to those recorded on comparable surface materials outside basins. Also it has been shown that it is possible that basin response effects are also important at higher frequencies (Davis et al., 2000), or can affect structures having higher fundamental mode frequencies. Where the basin effects "end" and the local ground response "begins" is ambiguous. In the present work local ground response effects are defined as representing the "one-dimensional ground response of the soil column" (which in practice is often modeled using only the upper several hundred meters), and basin effects as producing ground motions that deviate from the predictions of the one-dimensional models as a result of relatively complex wave propagation in basins. The term "one-dimensional ground response" refers to the modification of vertically propagating waves by sediment layers. It is one of the two processes that contribute to the motion amplification at sediment sites as previously mentioned. The soil

properties most commonly utilized for horizontal ground motions include a profile of small strain with shear wave velocity (V_S), relationships for the variation of the normalized shear modulus (G/G_{max}), hysteretic soil damping (ζ) with shear strain (γ). Ground response analysis models solve the equation of motion for one-dimensional wave propagation. The distinguishing feature among the analysis routines is the soil material model, for which three general categories exist: equivalent linear and nonlinear models for one horizontal direction of shaking and nonlinear models for multiple directions of shaking. The most popular code in the first category is SHAKE91, developed by (Idriss and Sun, 1992).

The term "local site effects" refers to the influence of relatively shallow (top 30 m) soil materials on (nearly) vertically propagating waves. These effects are ideally modeled using the full soil profile, but for deep alluvial basins, the modeling domain generally does not extend beyond depths of 100-200 m. A means of characterizing the top 30 m layers is via the average shear wave velocity V_S which is defined as (using the total distance traveled over the travel time through each layer):

$$V_S = \frac{\sum_{i=1}^{n} d_i}{\sum_{i=1}^{n} \frac{d_i}{V_{Si}}}$$

Finally the term "surface topography" is self explanatory.

A discussion of factors influencing ground motion can be found in Stewart et al. (2001). The traditional way of analyzing structures using a fixed base model does not allow for analyses that incorporate the factors that influence the ground motion. The need for a sophisticated analysis tool that enables the modeling of the factors contributing to complex site effects and allows for accounting for site effects in structural analysis is evident.

1.2.2 Soil-foundation-structure interaction

During the shaking of an earthquake, seismic waves are transmitted from the fault rupture through the soil to a structure. The wave motion of the soil excites the structure which in turn modifies the input motion by its movement relative to the ground. This interaction is termed "soil-foundation-structure interaction" (SFSI) or sometimes simply "soil-structure interaction" (SSI) even though accounting for the foundation type is of equally critical importance as that of the modeling of the structures. Depending on the soil properties, the type of the foundation and the seismic waves, the response of a structure can be quite different from the case when the supporting system is rigid. This interaction is generally considered to be favorable in earthquake engineering design, meaning that the design demands as computed by a fixed base analysis are assumed to be higher than the demands due to a SFSI analysis. Regardless of its effect, SFSI imposes additional complication in the analysis. It is characterized by complex wave propagation phenomena which can be summarized as follows:

- The influence of non-vertically incident seismic waves. Complicated coupling phenomena may occur for non-vertically incident body waves or for surface waves, since those tend to cause rotation of the foundation as well as translation. The rotational component is neglected if SFSI is neglected but can be very important for tall buildings.

- The dissipation of energy from the superstructure is important for characterizing the building response. In the case when interaction is not neglected, the semi-infinite soil medium acts as a sink because the energy is dissipated by geometrical radiation of energy.

- The configuration of the superstructure and its foundation can be important. For example torsional vibration may be induced by horizontal excitation if the structure is not symmetrical.

- The influence of surrounding buildings may also be significant. The vibration of the nearby foundations can be thought as additional wave sources. Therefore, in densely constructed urban areas, motion of a particular foundation may be amplified or attenuated by the existence of neighboring structures. This effect can be more significant when the nearby structures are heavier than the one under consideration.

- A flexible support may allow larger relative movements between the heavier structural frames which can result in high localized stresses.

In general two mechanisms of interaction (Stewart et al., 1999a,b) take place between the soil, foundation and structure:

- Inertial Interaction: Inertia developed in the structure due to its own vibrations gives rise to base shear and moment, which in turn cause displacements of the foundation relative to the free-field. Frequency dependent foundation impedance functions describe the flexibility of the foundation support as well as the damping associated with foundation-soil interaction.

- Kinematic Interaction: The presence of stiff foundation elements on or in the soil cause foundation motions to deviate from free-field motions as a result of ground motion incoherence, wave inclination, or foundation embedment. Kinematic effects are described by a frequency dependent transfer function relating the free-field motion to the motion that would occur on the base slab if the slab and structure were massless.

Inertial interaction is the most important effect when foundations do not have large rigid base slabs or deep embedment. For building structures, inertial interaction is the most common case and the focus of this work. A system commonly employed for the simplified analysis of inertial interaction is shown in Fig. 1.1. The impedance function for the foundation and soil, is represented by $\bar{k}_u$ and $\bar{k}_\theta$ and perhaps a coupling spring. Simplified impedance function solutions are available for rigid circular disk foundations

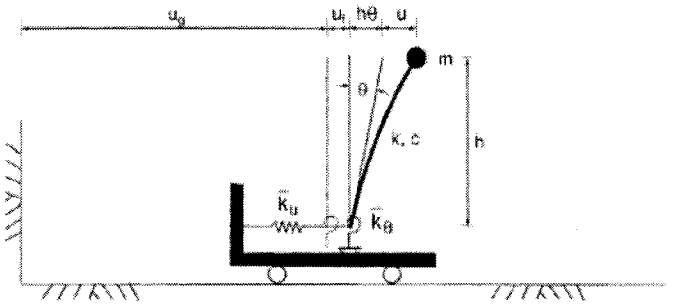

Figure 1.1. Simplified model for analysis of inertial interaction (Stewart et al., 1999a)

mounted or embedded in a uniform, viscoelastic half-space. Analytical procedures are available for the computation of impedance functions for rigid foundations, many of which are summarized in Luco (1980) and Roeset (1980). There are also important effects due to nonuniform soil profiles, embedded foundations, noncircular foundation shapes, flexible foundations, and piles or piers beneath the base slab that must be taken into account, and that are also addressed in the literature.

For the effects of inertial SFSI on structures Veletsos and Meek (1974) found that the seismically induced deformations of a single degree of freedom system with a surface foundation can be accurately represented by an equivalent fixed base single degree of freedom system with period $\tilde{T}$ and damping ratio $\tilde{\zeta}$, which are called the flexible-base properties of the system. The flexible-base period as calculated by Veletsos and Meek (1974) is given by:

$$\frac{\tilde{T}}{T} = \sqrt{1 + \frac{k}{k_u} + \frac{kh^2}{k_\theta}} \tag{1.1}$$

where T is the fixed-base period of the structure in Fig. ??. The flexible-base damping ratio as calculated by Veletsos and Nair (1975) is given by:

$$\tilde{\zeta} = \tilde{\zeta}_0 + \frac{\zeta}{(\tilde{T}/T)^3} \tag{1.2}$$

A series of analysis procedures and system identification techniques for evaluating inertial SSI effects on seismic structural response is given in Stewart et al. (1999a,b).

Kinematic interaction occurs in the form of base-slab averaging or in the case of deep foundations. Base-slab averaging results from incoherent or incident wave fields. Motions on surface foundations are modified relative to the free field when incident waves impinge on the foundation with an oblique angle, or when the incident wave is incoherent. The first case is referred to as the wave passage effect and the second case as the ground motion incoherence effect. In the presence of these wave fields, translational base-slab motions are reduced relative to the free field, and rotational motions are introduced. The rotational motions include rocking in the presence of inclined SV waves, P waves, and Raleigh waves, and torsion in the presence of SH waves or Love waves. The reductions of base-slab translation, and the introduction of torsion and rocking, are all effects that tend to become more significant with increasing frequency. The frequency dependence of these effects is associated with the increased effective size of the foundation relative to the seismic wavelengths at higher frequencies. In addition, ground motions are more incoherent at higher frequencies. The seismic response of a pile-supported foundation differs from that of a surface foundation due to the increased stiffness of the pile-soil system as compared to soil only, and due to the scattering of seismic waves from the piles. Most theoretical studies of kinematic effects associated with pile-soil interaction have been performed for single piles or pile groups with a rigid cap not in contact with the ground.

In brief conclusion, SFSI is a problem widely investigated both theoretically and experimentally by many researchers, and fairly well understood. Closed form solutions are provided for many idealized cases. However the need for an analysis tool that enables complex 3D analysis capabilities, thus setting the limitations of analytic potential into the sophistication and detail of modeling of the soil, foundation and structure is still present, to address more complex problems such as nonlinear SFSI modeling for detailed system response analyses and the structural design of foundation elements.

1.3 Projects similar to SPUR

In the context of simulating the seismic behavior of an urban region researchers have conducted several investigations. A difficulty which researchers often encounter is the scale of the model which is hundreds of millions or billions of degrees of freedom, thus presenting enormous computational demands. Another difficulty is the inability to represent the structural inventory, which mainly arises from lack of detailed enough information about the buildings, the foundation and the soil. Research projects with aims similar to those of the SPUR project are briefly summarized next.

1.3.1 Macro-Micro analysis for prediction of strong motion distribution in metropolis

In this project by Ichimura and Hori (2000), the researchers utilized parallel computing to efficiently perform the simulation of a 300 m $\times$ 300 m $\times$ 60 m subregion of the Roppongi Area, in Tokyo, Japan; which contained about 150 buildings. The soil shear wave velocity profiles V_S ranged from 120 m to 600 m. The structural data required where available from Japan's GIS, and for each building an equivalent MDOF system was constructed, according to Chopra (2001). Their methodology for analyzing their virtual city was as follows:

1. Carry out the macro-analysis for the geological model, and obtain the input for the micro-analysis.

2. Carry out the micro-analysis for the underground structure model, and obtain the displacement distribution on the ground surface as time data series.

3. Compute the structure responses by inputting the strong motion to the MDOF systems and applying the modal analysis.

The disadvantage of this methodology is that the analysis of the structures was decoupled from the soil simulation. Consequently the effects of SFSI and the interactions between structures were not included.

1.3.2 Site-city interaction in urban areas

This work by Semblat et al. (2004) treated the structural inventory of an urban area as an additional earthquake source that modifies the free-field ground motion contents. The problem was investigated at the scale of a basin and a whole city to examine the potential influence of "city-site interaction" on site response and the resulting ground motion. A two-dimensional model was used to represent an actual alluvial valley in Nice, France. The boundary element method was used to compute the wave propagation in the frequency domain. The excitation was limited to a vertically propagating SH wave and 33 buildings where accounted for. The scale of the model was 2 km. This work was limited to a specific type of wave propagation problem and thus did not represent the three-dimensional nature of the wave propagation and the soil-structure interaction problem.

1.3.3 HAZUS methodology for damage prediction in an urban area

HAZUS (FEMA, 2008) is a methodology developed by the Federal Emergency Management Agency and the National Institute of Building Sciences for estimating earthquake losses, intended for local, regional, or state officials contemplating an earthquake loss study. The methodology has been pilot tested in Portland, Oregon and Boston, Massachusetts and calibrated with data from Northridge, Loma Prieta and other earthquakes. The methodology generates an estimate of the consequences to a city or region of a "scenario earthquake" – that is, an earthquake with a specified magnitude and location. The resulting "loss estimate" generally describes the scale and extent of damage and disruption that may result from potential earthquakes. The methodology is based on simple statistical approaches for the estimation of seismic performance.

1.4 Organization of the dissertation

The dissertation is organized as follows:

- Chapter 2 presents the methodology selected for providing effective seismic input in soil domains and provides a theoretical overview of the method. Issues of explicit time integration are discussed. The presentation of the algorithms and software design that lead in the development of a parallel application for the analysis of soil domains using explicit finite elements follow next. The chapter concludes with a scalability study of the performance of the parallel application. Almost linear weak scalability is demonstrated.

- Chapter 3 contains an application of the simulation software for the study of the response of a soil region in downtown Los Angeles under two different excitations: a low frequency one and a broadband. Also the effect of softening the soil profile is studied. The area under study is in very close proximity with the Puente Hills fault.

- Chapter 4 presents the use of mixed explicit-implicit time integration algorithms for analyzing coupled domains of soil and structures. The modification of an existing algorithm is developed, with an extension for nonlinear systems in the implicit partition. The algorithms are presented together with an efficient software design that allows for maintaining the scalability and performance of the explicit application described in Chapter 2.

- Chapter 5 utilizes the tools developed in the previous two chapters in order to analyze the seismic performance of the structural inventory of the same region previously mentioned. Simplified structural models are used for the simulations.

- Chapter 6 contains conclusions and points to certain future directions.

Chapter 2

Modeling of soil domains

This chapter describes the Domain Reduction Method as a means for providing effective seismic input in soil domains. The theoretical and implementation aspects of the method are examined. A brief discussion of the time integration methods for wave propagation through soil domains follows. The software design and implementations of a set of classes that are used together with components of a large software framework to form a custom, very scalable and high performance parallel application used for the analysis of soil domains are presented next. The chapter concludes with the presentation of scalability results for this application.

2.1 Theory background related to modeling of soil domains

2.1.1 Introduction

In the recent years, a tremendous development has occurred in the simulation of earthquake motions using physics based 3D models in seismic regions. Numerical modeling methods for anelastic wave propagation that take into consideration the earthquake source, propagation path, and local site effects have been significantly advanced by many

researchers. Several types of these methods exist, such as boundary-element and discrete wave-number methods, suitable for moderately sized problems with relatively simple geometrical and geological conditions, finite difference methods (Olsen, 2001; Graves, 1993) and finite element methods (Aagard et al., 2001; Bao et al., 2001). The latter are better suited for large-sized problems involving realistic basin models with material heterogeneity because of the ability to define arbitrary geometries and material properties. Common difficulties that researchers encounter when utilizing finite difference methods are restrictive simplifications and approximations such as limiting the maximum frequency or lowest shear wave velocities that are considered. The main reason for this is that most methods currently in use for large-sized problems are based upon uniform structural grids.

Finite elements are more flexible since they can better tailor the mesh size to local wavelengths of the propagating waves. A common feature of both the finite difference and the finite element methods is that the ground motions near the causative fault and those along the propagation path and in the region of interest are all calculated simultaneously, using a single computational model that encompasses the entire geological structure. This allows for incorporating source, propagation path and local site effects in the same analysis. This approach of using a single model works well in many cases, however when the source is far from the region of interest or when it is desired to analyze the region's structural inventory, the exceedingly large size of the model renders any such method ineffective.

One method that uses a finite element formulation but avoids the need to represent accurately the geometric and material properties of the whole region within a single computational model, consists in subdividing the problem in two sequential computational phases. First, a background structure is considered from which the localized features have been removed and the corresponding ground motion is computed. The computational grid required for this step is as dictated by the softest material under consideration and the desired level of accuracy for the representation of a certain frequency range. The ground motion computed from this first step is used to compute a set of localized equivalent earth-

quake forces which are then applied as input in a computational domain that represents the region of interest (a subdomain of the first step's domain) with all its local features and structural inventory if desired. This method has many advantages for the problem of detailed soil-foundation-structure interaction analysis, was developed by Bielak et al. (2003a,b) and is named the Domain Reduction Method (DRM).

In addition to the increased numerical method availability, major progress in the computer science fields of architecture and systems has enabled the usage of large computing clusters and supercomputers with very high performance and capacities that allow the use of many processors for a simulation. This has consequently increased tremendously the potential size of the finite element model in addition to the speed of the numerical solution of the resulting systems of equations. Parallel computing is now widely available and finite elements codes have closely followed these evolutions. This is another very important reason that motivates the usage of a finite element based methodology as the DRM.

2.1.2 The domain reduction method (DRM)

The Domain Reduction Method (DRM) developed by Bielak et al. (2003a,b), is a two-step, finite element based methodology for modeling earthquake ground motion in media characterized by high heterogeneity and significant contrast in wavelengths. The problem that the method treats is a semi-infinite seismic region, that includes the fault-system, the wave propagation path and localized geological features in three dimensions. Since the localized geological features require higher frequency resolution than the large scale wave propagation path, the DRM separates the problem into two sub-problems. The first solves the wave propagation of the seismic excitation from the fault system to the region of interest (ROI), and the second sub-problem provides the detailed analysis of the smaller subdomain that comprises the ROI under the equivalent excitation that results from the wave propagation of the earthquake excitation from the source.

Fig. 2.1 shows a domain of interest. The interface layer Γ divides the domain into

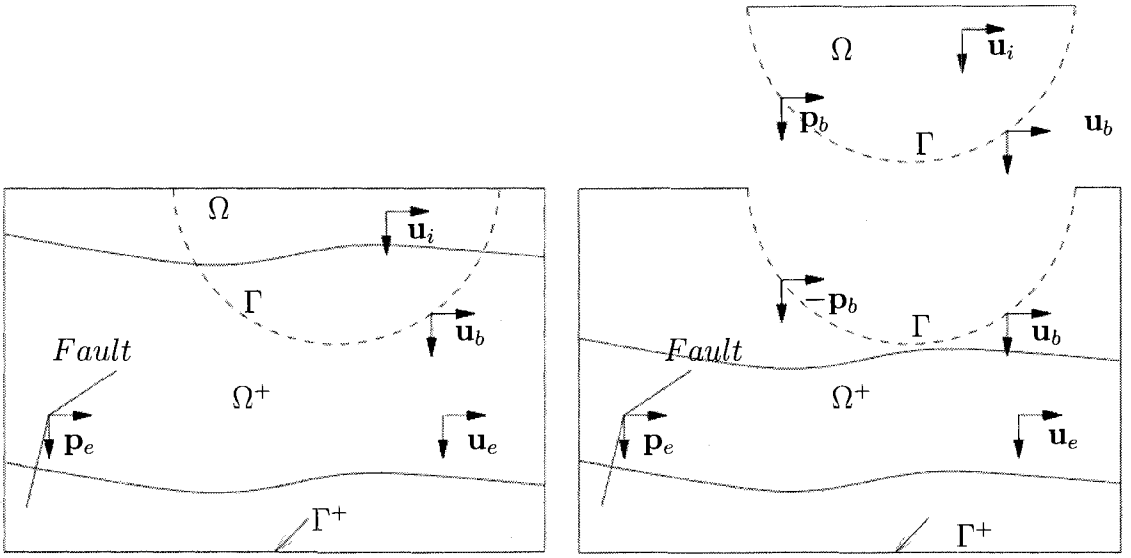

Figure 2.1. Truncated seismic region. Outer boundary Γ^+ restricts computations to a finite domain; fictitious interface Γ divides region into two subdomains: Ω^+ which includes the seismic source represented by nodal forces P_e, and Ω which contains the localized geological features (Bielak et al., 2003a)

two parts: Ω which contains the ROI, and Ω^+ which is the semi-infinite half-space that includes the fault system. The exterior boundary Γ^+ allows for the modeling of the semi-infinite domain as finite. The entire system $\Omega^+ \cup \Omega$ is governed by Navier's equation's of elastodynamics which upon finite element spatial discretization can be written for the entire domain as:

$$\begin{bmatrix} \mathbf{M}_{ii}^{\Omega} & \mathbf{M}_{ib}^{\Omega} & \mathbf{0} \\ \mathbf{M}_{bi}^{\Omega} & \mathbf{M}_{bb}^{\Omega} + \mathbf{M}_{bb}^{\Omega^+} & \mathbf{M}_{be}^{\Omega^+} \\ \mathbf{0} & \mathbf{M}_{eb}^{\Omega^+} & \mathbf{M}_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{u}}_i \\ \ddot{\mathbf{u}}_b \\ \ddot{\mathbf{u}}_e \end{Bmatrix} + \begin{bmatrix} \mathbf{C}_{ii}^{\Omega} & \mathbf{C}_{ib}^{\Omega} & \mathbf{0} \\ \mathbf{C}_{bi}^{\Omega} & \mathbf{C}_{bb}^{\Omega} + \mathbf{C}_{bb}^{\Omega^+} & \mathbf{C}_{be}^{\Omega^+} \\ \mathbf{0} & \mathbf{C}_{eb}^{\Omega^+} & \mathbf{C}_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}_i \\ \dot{\mathbf{u}}_b \\ \dot{\mathbf{u}}_e \end{Bmatrix}$$
$$+ \begin{bmatrix} \mathbf{K}_{ii}^{\Omega} & \mathbf{K}_{ib}^{\Omega} & \mathbf{0} \\ \mathbf{K}_{bi}^{\Omega} & \mathbf{K}_{bb}^{\Omega} + \mathbf{K}_{bb}^{\Omega^+} & \mathbf{K}_{be}^{\Omega^+} \\ \mathbf{0} & \mathbf{K}_{eb}^{\Omega^+} & \mathbf{K}_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_i \\ \mathbf{u}_b \\ \mathbf{u}_e \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{p}_e \end{Bmatrix} \tag{2.1}$$

Applying the partitioning that the interface Γ defines in Fig. 2.1, the Eq. (2.1) can be rewritten as:

$$\begin{bmatrix} \mathbf{M}_{ii}^{\Omega} & \mathbf{M}_{ib}^{\Omega} \\ \mathbf{M}_{bi}^{\Omega} & \mathbf{M}_{bb}^{\Omega} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{u}}_i \\ \ddot{\mathbf{u}}_b \end{Bmatrix} + \begin{bmatrix} \mathbf{C}_{ii}^{\Omega} & \mathbf{C}_{ib}^{\Omega} \\ \mathbf{C}_{bi}^{\Omega} & \mathbf{C}_{bb}^{\Omega} \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}_i \\ \dot{\mathbf{u}}_b \end{Bmatrix} + \begin{bmatrix} \mathbf{K}_{ii}^{\Omega} & \mathbf{K}_{ib}^{\Omega} \\ \mathbf{K}_{bi}^{\Omega} & \mathbf{K}_{bb}^{\Omega} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_i \\ \mathbf{u}_b \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{p}_b \end{Bmatrix} \tag{2.2}$$

and

$$\begin{bmatrix} \mathbf{M}_{bb}^{\Omega^+} & \mathbf{M}_{be}^{\Omega^+} \\ \mathbf{M}_{eb}^{\Omega^+} & \mathbf{M}_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{u}}_b \\ \ddot{\mathbf{u}}_e \end{Bmatrix} + \begin{bmatrix} \mathbf{C}_{bb}^{\Omega^+} & \mathbf{C}_{be}^{\Omega^+} \\ \mathbf{C}_{eb}^{\Omega^+} & \mathbf{C}_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}_b \\ \dot{\mathbf{u}}_e \end{Bmatrix} + \begin{bmatrix} \mathbf{K}_{bb}^{\Omega^+} & \mathbf{K}_{be}^{\Omega^+} \\ \mathbf{K}_{eb}^{\Omega^+} & \mathbf{K}_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_b \\ \mathbf{u}_e \end{Bmatrix} = \begin{Bmatrix} -\mathbf{p}_b \\ \mathbf{p}_e \end{Bmatrix} \tag{2.3}$$

In the above equations the matrices $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{K}$ denote mass, damping and stiffness matrices respectively, the subscripts e, b and i denote the nodes in the exterior domain, interface and interior domain. The superscripts Ω^+ and Ω refer to the domain over which the matrices are defined. To transfer the excitation from the causative fault to the ROI an auxiliary problem is defined in which for the exterior domain, the materials and the source remain identical to those of the original problem but the interior domain is substituted by a simpler one denoted as Ω_0, such that the entire system now defined as $\Omega^+ \cup \Omega_0$ is easier to be solved. Now rewriting Eq. (2.3) for Ω^+:

$$\begin{bmatrix} \mathbf{M}_{bb}^{\Omega^+} & \mathbf{M}_{be}^{\Omega^+} \\ \mathbf{M}_{eb}^{\Omega^+} & \mathbf{M}_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{u}}_b^0 \\ \ddot{\mathbf{u}}_e^0 \end{Bmatrix} + \begin{bmatrix} \mathbf{C}_{bb}^{\Omega^+} & \mathbf{C}_{be}^{\Omega^+} \\ \mathbf{C}_{eb}^{\Omega^+} & \mathbf{C}_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}_b^0 \\ \dot{\mathbf{u}}_e^0 \end{Bmatrix} + \begin{bmatrix} \mathbf{K}_{bb}^{\Omega^+} & \mathbf{K}_{be}^{\Omega^+} \\ \mathbf{K}_{eb}^{\Omega^+} & \mathbf{K}_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_b^0 \\ \mathbf{u}_e^0 \end{Bmatrix} = \begin{Bmatrix} -\mathbf{p}_b^0 \\ \mathbf{p}_e \end{Bmatrix} \tag{2.4}$$

The matrices $\mathbf{M}$, $\mathbf{C}$ and $\mathbf{K}$ are identical to those in Eq. (2.3) and so is $\mathbf{p}_e$. The second component of Eq. (2.4) can be expressed in terms of $\mathbf{p}_e$ as:

$$\mathbf{p}_e = \mathbf{M}_{eb}^{\Omega^+} \ddot{\mathbf{u}}_b^0 + \mathbf{M}_{ee}^{\Omega^+} \ddot{\mathbf{u}}_e^0 + \mathbf{C}_{eb}^{\Omega^+} \dot{\mathbf{u}}_b^0 + \mathbf{C}_{ee}^{\Omega^+} \dot{\mathbf{u}}_e^0 + \mathbf{K}_{eb}^{\Omega^+} \mathbf{u}_b^0 + \mathbf{K}_{ee}^{\Omega^+} \mathbf{u}_e^0 \tag{2.5}$$

By substitution of Eq. (2.5) into Eq. (2.1), a direct solution for the unknown wavefield can be obtained. This formulation though offers no computational advantage over a traditional

finite element formulation of the problem, since it requires the free field wave $\mathbf{u}_e^0$ stored throughout Ω^+.

To devise an efficient scheme the DRM (Bielak et al. (2003a,b)) proceeds with a variable transformation such that the total wavefield $\mathbf{u}_e$ is expressed as the summation of the free field due to the background structure $\mathbf{u}_e^0$ and a residual displacement due to the ROI which is the relative displacement field with respect to the free field $\mathbf{w}_e$:

$$\mathbf{u}_e = \mathbf{u}_e^0 + \mathbf{w}_e \tag{2.6}$$

Substitution of the above into Eq. (2.1) yields:

$$\begin{aligned}
&\begin{bmatrix} \mathbf{M}_{ii}^{\Omega} & \mathbf{M}_{ib}^{\Omega} & \mathbf{0} \\ \mathbf{M}_{bi}^{\Omega} & \mathbf{M}_{bb}^{\Omega} + \mathbf{M}_{bb}^{\Omega^+} & \mathbf{M}_{be}^{\Omega^+} \\ \mathbf{0} & \mathbf{M}_{eb}^{\Omega^+} & \mathbf{M}_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{u}}_i \\ \ddot{\mathbf{u}}_b \\ \ddot{\mathbf{w}}_e \end{Bmatrix} + \begin{bmatrix} \mathbf{C}_{ii}^{\Omega} & \mathbf{C}_{ib}^{\Omega} & \mathbf{0} \\ \mathbf{C}_{bi}^{\Omega} & \mathbf{C}_{bb}^{\Omega} + \mathbf{C}_{bb}^{\Omega^+} & \mathbf{C}_{be}^{\Omega^+} \\ \mathbf{0} & \mathbf{C}_{eb}^{\Omega^+} & \mathbf{C}_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}_i \\ \dot{\mathbf{u}}_b \\ \dot{\mathbf{w}}_e \end{Bmatrix} \\
&\qquad + \begin{bmatrix} \mathbf{K}_{ii}^{\Omega} & \mathbf{K}_{ib}^{\Omega} & \mathbf{0} \\ \mathbf{K}_{bi}^{\Omega} & \mathbf{K}_{bb}^{\Omega} + \mathbf{K}_{bb}^{\Omega^+} & \mathbf{K}_{be}^{\Omega^+} \\ \mathbf{0} & \mathbf{K}_{eb}^{\Omega^+} & \mathbf{K}_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_i \\ \mathbf{u}_b \\ \mathbf{w}_e \end{Bmatrix} \\
&\qquad = \begin{Bmatrix} \mathbf{0} \\ -\mathbf{M}_{be}^{\Omega^+} \ddot{\mathbf{u}}_e^0 - \mathbf{C}_{be}^{\Omega^+} \dot{\mathbf{u}}_e^0 - \mathbf{K}_{be}^{\Omega^+} \mathbf{u}_e^0 \\ \mathbf{p}_e - \mathbf{M}_{ee}^{\Omega^+} \ddot{\mathbf{u}}_e^0 - \mathbf{C}_{ee}^{\Omega^+} \dot{\mathbf{u}}_e^0 - \mathbf{K}_{ee}^{\Omega^+} \mathbf{u}_e^0 \end{Bmatrix}
\end{aligned} \tag{2.7}$$

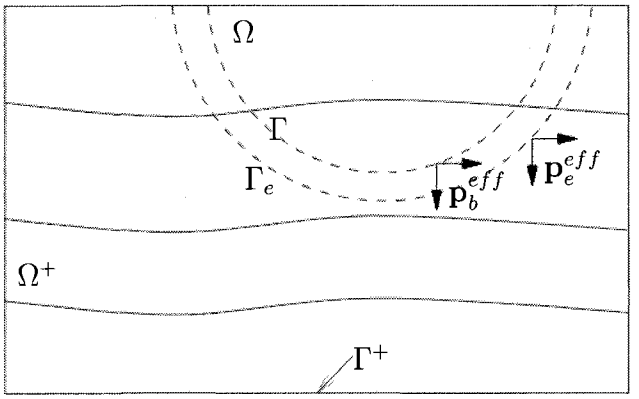

Figure 2.2. Seismic region with the two neighboring surfaces Γ and Γ_e where the effective forces $\mathbf{P}^{eff}$ defined by Eq. (2.9) are to be applied (Bielak et al., 2003a)

Substitution for $\mathbf{p}_e$ from Eq. (2.5) into Eq. (2.7) yields:

$$\begin{bmatrix} \mathbf{M}_{ii}^{\Omega} & \mathbf{M}_{ib}^{\Omega} & \mathbf{0} \\ \mathbf{M}_{bi}^{\Omega} & \mathbf{M}_{bb}^{\Omega}+\mathbf{M}_{bb}^{\Omega^+} & \mathbf{M}_{be}^{\Omega^+} \\ \mathbf{0} & \mathbf{M}_{eb}^{\Omega^+} & \mathbf{M}_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{u}}_i \\ \ddot{\mathbf{u}}_b \\ \ddot{\mathbf{w}}_e \end{Bmatrix} + \begin{bmatrix} \mathbf{C}_{ii}^{\Omega} & \mathbf{C}_{ib}^{\Omega} & \mathbf{0} \\ \mathbf{C}_{bi}^{\Omega} & \mathbf{C}_{bb}^{\Omega}+\mathbf{C}_{bb}^{\Omega^+} & \mathbf{C}_{be}^{\Omega^+} \\ \mathbf{0} & \mathbf{C}_{eb}^{\Omega^+} & \mathbf{C}_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}_i \\ \dot{\mathbf{u}}_b \\ \dot{\mathbf{w}}_e \end{Bmatrix}$$
$$+ \begin{bmatrix} \mathbf{K}_{ii}^{\Omega} & \mathbf{K}_{ib}^{\Omega} & \mathbf{0} \\ \mathbf{K}_{bi}^{\Omega} & \mathbf{K}_{bb}^{\Omega}+\mathbf{K}_{bb}^{\Omega^+} & \mathbf{K}_{be}^{\Omega^+} \\ \mathbf{0} & \mathbf{K}_{eb}^{\Omega^+} & \mathbf{K}_{ee}^{\Omega^+} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_i \\ \mathbf{u}_b \\ \mathbf{w}_e \end{Bmatrix} \tag{2.8}$$
$$= \begin{Bmatrix} \mathbf{0} \\ -\mathbf{M}_{be}^{\Omega^+}\ddot{\mathbf{u}}_e^0 - \mathbf{C}_{be}^{\Omega^+}\dot{\mathbf{u}}_e^0 - \mathbf{K}_{be}^{\Omega^+}\mathbf{u}_e^0 \\ \mathbf{M}_{eb}^{\Omega^+}\ddot{\mathbf{u}}_b^0 + \mathbf{C}_{eb}^{\Omega^+}\dot{\mathbf{u}}_b^0 + \mathbf{K}_{eb}^{\Omega^+}\mathbf{u}_b^0 \end{Bmatrix}$$

By means of these substitutions the seismic forces $\mathbf{p}_e$ have been transferred into Ω and replaced by the effective seismic forces $\mathbf{p}^{eff}$ as shown in Fig. 2.2:

$$\mathbf{p}^{eff} = \begin{Bmatrix} \mathbf{p}_i^{eff} \\ \mathbf{p}_b^{eff} \\ \mathbf{p}_e^{eff} \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ -\mathbf{M}_{be}^{\Omega^+}\ddot{\mathbf{u}}_e^0 - \mathbf{C}_{be}^{\Omega^+}\dot{\mathbf{u}}_e^0 - \mathbf{K}_{be}^{\Omega^+}\mathbf{u}_e^0 \\ \mathbf{M}_{eb}^{\Omega^+}\ddot{\mathbf{u}}_b^0 + \mathbf{C}_{eb}^{\Omega^+}\dot{\mathbf{u}}_b^0 + \mathbf{K}_{eb}^{\Omega^+}\dot{\mathbf{u}}_b^0 \end{Bmatrix} \tag{2.9}$$

The main advantage of the DRM is the separation of the problem into two steps. Transfer of the excitation from the causative fault to the ROI, and the solution of the

problem within the ROI. For the second step the wave field is required only at the single element layer interface Γ_e-Γ, and this is the advantage over the traditional one step approach. For the regional simulations, the equations of motion Eq. (2.8) within the ROI (Ω) must be solved. Thus it is required to set up and apply Eq. (2.9) to the subdomain under consideration. This implies significant computational cost. The design and implementation of a code that targets solution of soil subdomains using explicit time integration and the DRM method is presented later in this chapter.

2.1.3 Absorbing boundary conditions

In the previous discussion, there is an implicit requirement for the conversion of the semi-infinite domain to a finite one, thus allow for the use of finite elements. This requires that an absorbing boundary condition to be used on Γ^+, to prevent energy from reflecting back to the ROI. The absorbing boundary chosen is the Lysmer absorbing boundary presented by Lysmer and Kuhlemeyer (1969), which is a local boundary condition chosen because it provides a simple and efficient numerical procedure. That is opposed to global absorbing boundary conditions that are more accurate and computationally expensive due to the full coupling in space and time. This boundary absorbs the propagating waves in such a way that no outgoing waves from within the ROI are reflected back to the interior domain.

The presentation next follows Zhang et al. (2008). The one-dimensional vertical shear wave propagation equation is:

$$\frac{\partial^2 u(x,t)}{\partial t^2} = V_S \frac{\partial^2 u(x,t)}{\partial x^2} \tag{2.10}$$

where u denotes the soil particle displacement perpendicular to wave propagation direction and V_S is the soil shear wave velocity. The solution of the above is of the form:

$$u(x,t) = u_r(t - \frac{x}{V_S}) + u_i(t + \frac{x}{V_S}) \tag{2.11}$$

where $u_r(\dots)$ and $u_i(\dots)$ are functions of $(t - x/V_S)$ and $(t + x/V_S)$, respectively. The term u_r represents the wave traveling at velocity V_S in the positive x-direction, while u_i represents the wave traveling at the same speed in the negative x-direction. Thus, u_i is the incident to the computational domain wave and u_r is the reflected wave. Taking the partial derivative with respect to time of both sides of the above equation and multiplying by ρV_S gives:

$$\rho V_S \frac{\partial u(x,t)}{\partial t} = \rho V_S u_r'(t - \frac{x}{V_S}) + \rho V_S u_i'(t + \frac{x}{V_S}) \tag{2.12}$$

where the prime denotes differentiation with respect to the prime argument of the function. The uniaxial, linear elastic shear stress - shear strain relation is given by:

$$\tau(x,t) = G\frac{\partial u(x,t)}{\partial x} = -\frac{G}{V_S}u_r'(t - \frac{x}{V_S}) + \frac{G}{V_S}u_i'(t + \frac{x}{V_S}) \tag{2.13}$$

which using Eq. (2.12) and also that $V_S = \sqrt{G/\rho}$ can be rewritten as:

$$\tau(x,t) = -\rho V_S \frac{\partial u(x,t)}{\partial t} + 2\rho V_S u_i'(t + \frac{x}{V_S}) \tag{2.14}$$

The term $\partial u(x,t)/\partial t$ is the velocity of the total soil particle motion and the term $u_i'(t + x/V_S$ is the velocity of the incident motion. Therefore the first term of Eq. (2.14) is the equivalent of a force (per unit area) generated by a dashpot of coefficient ρV_s and the second term is equivalent to the force (per unit area) proportional to the velocity of the incident wave. Thus the soil surrounding the finite soil domain of our interest can be replaced by a dashpot, and a force for the equivalent seismic input.

The equivalent seismic input is provided by the DRM by the means we saw previously and the dashpots whose coefficients are shown in Fig. 2.3 are calibrated as follows:

$$\begin{aligned} C_P &= \rho V_S \\ C_N &= \rho V_P \\ f_{P_1} &= C_P \dot{u}_{P_1} \\ f_{P_2} &= C_P \dot{u}_{P_2} \\ f_N &= C_N \dot{u}_N \end{aligned} \tag{2.15}$$

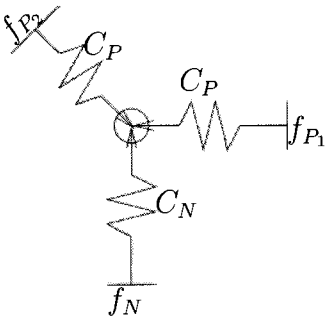

Figure 2.3. Lysmer one-dimensional absorbing boundaries

where ρ is the soil density, C_P and C_N are the damping coefficients for the tangential and normal directions respectively and f_{P_1},f_{P_2} and f_N are the forces per unit area generated by the dampers on the two tangential and the normal to the element's longitudinal axis plane. One question that can be raised here is that the energy absorption does not only depend on the material properties but also depends on the frequency contents of the wave field. The choice of the Lysmer absorbing boundary is dictated by the computational efficiency and ease of implementation.

2.2 Explicit time integration schemes for the solution of equations of motion

A time integration method is required for the solution of the system of ordinary differential equations (ODE) of Eq. (2.8). Since the soil elements that will be used have linear elastic governing material laws, no iterations are required for their state determination. The methods that are best suited for the wave propagation problem using finite element discretization are explicit methods. Among those the focus is on the central difference method and an explicit variant of the Newmark β method, which will be the primary method used in this work. The great advantage of the explicit algorithms for

transient analysis lies in the fact that they can lead to the solution of the resulting system of equations:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.16}$$

without requiring the factorization of $\mathbf{A}$, when it is diagonal, simply by performing:

$$\mathbf{x}_i = (\frac{1}{\mathbf{A}_{ii}})\mathbf{b}_i \tag{2.17}$$

2.2.1 Central difference method

The central difference method is a second order accurate explicit scheme. It is well suited for the wave propagation problem because it does not require matrix factorization for the solution of the resulting system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ under certain conditions that make it diagonal. The general formulation of the central difference method is:

$$\left[\frac{\mathbf{M}}{(\Delta t)^2} + \frac{\mathbf{C}}{2\Delta t}\right]\{\mathbf{u}_{n+1}\} = \mathbf{p}_n^{ext} - \left[\frac{\mathbf{M}}{(\Delta t)^2} - \frac{\mathbf{C}}{2\Delta t}\right]\{\mathbf{u}_{n-1}\} - \left[\mathbf{K} - \frac{2\mathbf{M}}{(\Delta t)^2}\right]\{\mathbf{u}_n\} \tag{2.18}$$

however notice that it results in a system of equations Eq. (2.16) where:

$$\mathbf{A} = \left[\frac{\mathbf{M}}{(\Delta t)^2} + \frac{\mathbf{C}}{2\Delta t}\right] \tag{2.19}$$

$$\mathbf{x} = \mathbf{u}_{n+1} \tag{2.20}$$

and

$$\mathbf{b} = \mathbf{p}_n^{ext} - \left[\frac{\mathbf{M}}{(\Delta t)^2} - \frac{\mathbf{C}}{2\Delta t}\right]\{\mathbf{u}_{n-1}\} - \left[\mathbf{K} - \frac{2\mathbf{M}}{(\Delta t)^2}\right]\{\mathbf{u}_n\} \tag{2.21}$$

From Eq. (2.19) it is clear that in the general case of Rayleigh damping where the damping matrix is $\mathbf{C} = \alpha\mathbf{M} + \beta\mathbf{K}$, the matrix $\mathbf{A}$ is not diagonal and thus factorization is required. Using mass proportional Rayleigh damping: $\mathbf{C} = \alpha\mathbf{M}$, combined with lumped mass matrices ($\mathbf{M}_{ij} = 0$ if $i \neq j$), results in $\mathbf{A} = (1 + \alpha)\left[\frac{\mathbf{M}}{(\Delta t)^2}\right]$. This is an explicit scheme that does not require factorization and the solution can be obtained by Eq. (2.17). Another approach that also leads to avoiding the matrix factorization is by performing operator

splitting on the damping matrix and sending the off- diagonal terms to the right hand side and evaluating them at the previous time step. This approach works well when mass and stiffness proportional damping are to be used but results in decreasing of the accuracy of the method.

2.2.2 Alternative central difference method

This presentation follows from Cook et al. (1988). For the case of general Rayleigh damping but still using lumped mass matrices, an alternative formulation of the same method is used, one that assumes a half time step lag of the damping forces and results in the following expression:

$$\left[\frac{\mathbf{M}}{(\Delta t)^2}\right]\{\mathbf{u}_{n+1}\} = \mathbf{p}_n{}^{ext} - \mathbf{p}_n{}^{int} + \left[\frac{1}{(\Delta t)^2}\mathbf{M}\right]\left\{\mathbf{u}_n - \Delta t\dot{\mathbf{u}}_{n-1/2}\right\} - [\mathbf{C}]\left\{\dot{\mathbf{u}}_{n-1/2}\right\} \quad (2.22)$$

where:

$$\dot{\mathbf{u}}_{n-1/2} = \frac{1}{\Delta t}(\mathbf{u}_n - \mathbf{u}_{n-1}) \quad (2.23)$$

Eq. (2.22) can be rewritten as:

$$\begin{aligned}\left[\frac{\mathbf{M}}{(\Delta t)^2}\right]\{\mathbf{u}_{n+1}\} = \mathbf{p}_n{}^{ext} - \mathbf{p}_n{}^{int} &+ \left[\frac{2}{(\Delta t)^2}\mathbf{M} - \frac{1}{\Delta t}\mathbf{C}\right]\{\mathbf{u}_n\} \\ &- \left[\frac{1}{(\Delta t)^2}\mathbf{M} - \frac{1}{\Delta t}\mathbf{C}\right]\{\mathbf{u}_{n-1}\}\end{aligned} \quad (2.24)$$

The later formulation clearly for the case of lumped mass matrix allows for an explicit method according to Eq. (2.16) where:

$$\mathbf{A} = \left[\frac{\mathbf{M}}{(\Delta t)^2}\right] \quad (2.25)$$

$$\mathbf{x} = \mathbf{u}_{n+1} \quad (2.26)$$

and

$$\mathbf{b} = \mathbf{p}_n^{ext} - \mathbf{p}_n^{int} + \left[\frac{2}{(\Delta t)^2}\mathbf{M} - \frac{1}{\Delta t}\mathbf{C}\right]\{\mathbf{u}_n\} - \left[\frac{1}{(\Delta t)^2}\mathbf{M} - \frac{1}{\Delta t}\mathbf{C}\right]\{\mathbf{u}_{n-1}\} \quad (2.27)$$

the solution is very easily obtained by Eq. (2.17).

An alternative way of "diagonalizing" the central difference method would be to ignore the damping forces and use again a lumped mass matrix approach as suggested in Bathe (1996).

2.2.3 Newmark β explicit variant

This algorithm belongs to the general family of explicit predictor-corrector algorithms in Hughes et al. (1979):

$$\mathbf{M}\ddot{\mathbf{u}}_{n+1} + f(\tilde{\mathbf{u}}_{\mathbf{n+1}}, \tilde{\mathbf{u}}_{\mathbf{n+1}}) = \mathbf{p}_{\mathbf{n+1}}$$

Liu and Belytschko (1982) use an explicit predictor-corrector method that is based on the Newmark β-algorithm and is "compatible" with it:

$$\mathbf{M}\ddot{\mathbf{u}}_{n+1} + \mathbf{C}\tilde{\dot{\mathbf{u}}}_{n+1} + \mathbf{K}\tilde{\mathbf{u}}_{n+1} = \mathbf{p}_{n+1} \tag{2.28}$$

which leads to Eq. (2.18) with:

$$\mathbf{A} = \left[\frac{\mathbf{M}}{(\Delta t)^2}\right] \tag{2.29}$$

$$\mathbf{x} = \boldsymbol{\Delta}\mathbf{u}_{n+1} \tag{2.30}$$

and

$$\mathbf{b} = \mathbf{p}_{n+1}^{ext} - \mathbf{K}\tilde{\mathbf{u}}_{n+1} - \mathbf{C}\tilde{\dot{\mathbf{u}}}_{n+1} - \mathbf{M}\tilde{\ddot{\mathbf{u}}}_{n+1} \tag{2.31}$$

where:

$$\tilde{\mathbf{u}}_{n+1} = \mathbf{u}_n + \Delta t\dot{\mathbf{u}}_n + \frac{(\Delta t)^2}{2}(1-2\beta)\ddot{\mathbf{u}}_n \tag{2.32}$$

$$\tilde{\dot{\mathbf{u}}}_{n+1} = \dot{\mathbf{u}}_n + \Delta t(1-\gamma)\ddot{\mathbf{u}}_n \tag{2.33}$$

$$\tilde{\ddot{\mathbf{u}}}_{n+1} = -\frac{1}{\beta\Delta t}\dot{\mathbf{u}}_n + (1 - \frac{1}{2\beta})\ddot{\mathbf{u}}_n \tag{2.34}$$

This method is used for the simulations in this work.

2.2.4 Accuracy and stability of the time integration scheme

The term stability is used both for ODE initial value problems (IVP) and also with respect to numerical methods for their solution. First, a brief discussion about ODE solution stability is given. Assume a linear system of ODEs:

$$\begin{aligned} \dot{u}_1(t) &= a_{11}u_1(t) + a_{12}u_2(t) \\ \dot{u}_2(t) &= a_{21}u_1(t) + a_{22}u_2(t) \end{aligned} \tag{2.35}$$

which can be written as:

$$\dot{\mathbf{u}} = \mathbf{A}\mathbf{u}$$

Assuming $\mathbf{A}$ is invertible and with distinct eigenvalues, the sole fixed point ($\dot{\mathbf{u}} = \mathbf{0}$) is $\mathbf{u} = \mathbf{0}$. The general solution is:

$$\dot{\mathbf{u}} = \alpha_1 e^{\lambda_1 t}\mathbf{c_1} + \alpha_2 e^{\lambda_2 t}\mathbf{c_2}$$

where $\lambda_{1,2}$ and $\mathbf{c_{1,2}}$ are eigenvalues and eigenvectors of $\mathbf{A}$ respectively and $\alpha_{1,2}$ are constants due to the initial conditions. The stability of the system is:

- $\lambda_{1,2} \in \mathbb{R}$
  - $\lambda_{1,2} < 0$: $\lim_{t\to+\infty} \mathbf{u} = \mathbf{0} \Rightarrow stable$
  - $\lambda_{1,2} > 0$: $\lim_{t\to+\infty} \mathbf{u} = \infty \Rightarrow unstable$
  - $\lambda_1 < 0 < \lambda_2$:
    - If $\mathbf{u}(0) = \alpha\mathbf{c_1}$ then $\lim_{t\to+\infty} \mathbf{u} = \mathbf{0}$

* If $\mathbf{u}(0) = \alpha\mathbf{c_2}$ then $\lim_{t\to+\infty} \mathbf{u} = \infty$

unstable saddle

- $\lambda_{1,2} \in \mathbb{C}$: Then $\lambda = \mu \pm \nu i$ and assuming $\mathbf{u} \in \mathbb{R}^2$ then $\mathbf{u} = e^{\mu t}(\mathbf{d}_1 cos\nu t + \mathbf{d}_2 sin\nu t)$ where the $\mathbf{d}_{1,2}$ are formed by $\mathbf{A}$'s eigenvectors and the initial conditions.
  - $\mu > 0 \Rightarrow$ *unstable*
  - $\mu < 0 \Rightarrow$ *stable*
  - $\mu = 0 \Rightarrow$ *marginally stable*
  - If $\lambda_1 = \lambda_2$ the case is left to the literature.

In this work stability is used with respect to the numerical methods used for the solution of the system of ODE: Eq. (2.8). Excellent references on numerical methods for nonstiff and stiff problems are Hairer et al. (1993, 2004). The definition of stability for a general multistep numerical method is given in Hairer et al. (2004) for a general k-multistep method:

$$\alpha_k y_{m+k} + \cdots + a_0 y_m = h(\beta_k f_{m+k} + \cdots + \beta_0 f_m) \tag{2.36}$$

where $y' = f$. Applying it to the linearized autonomous system $y' = Jy$ we have:

$$\alpha_k y_{m+k} + \cdots + a_0 y_m = hJ(\beta_k y_{m+k} + \cdots + \beta_0 y_m) \tag{2.37}$$

For the coefficients of y_{m+i} with respect to an eigenvector v of J we have the same recurrence as Eq. (2.37) with J replaced by the corresponding eigenvalue λ.

$$(\alpha_k - \mu\beta_k)y_{m+k} + \cdots + (a_0 - \mu\beta_0)y_m = 0 \tag{2.38}$$

where $\mu = h\lambda$. To solve Eq. (2.38) Lagrange's method is used by setting $y_i = \xi^i$, and dividing with ξ^m throughout:

$$(\alpha_k - \mu\beta_k)\xi^k + \cdots + (a_0 - \mu\beta_0) = \rho(\xi) - \mu\sigma(\xi) = 0 \tag{2.39}$$

where $\rho(\xi) = \alpha_k\xi^k + \alpha_{k-1}\xi^{k-1} + \cdots + \alpha_0$ and $\sigma(\xi) = \beta_k\xi^k + \beta_{k-1}\xi^{k-1} + \cdots + \beta_0$.

Definition 1. *The set*

$$S = \left\{ \mu \in \mathbb{C}; \begin{array}{c} \text{all roots } \xi_j(\mu) \text{ of Eq. (2.39) satisfy } |\xi_j(\mu)| \leq 1, \\ \text{multiple roots satisfy } |\xi_j(\mu)| < 1 \end{array} \right\}$$

is called the stability domain or stability region or region of absolute stability of Method (2.36).

The stability and accuracy of the time integration schemes are presented next following Belytschko et al. (2000). For simplicity the analysis is on free vibration of a one-degree-of freedom system, hence all involved quantities are scalar:

$$M\ddot{u}_n + C\dot{u}_n + p_n^{int} = 0 \tag{2.40}$$

The central difference method uses:

$$\ddot{u}_n = \frac{1}{(\Delta t)^2}(u_{n+1} - 2u_n + u_{n-1}) \tag{2.41}$$

$$\dot{u}_n = \frac{1}{2\Delta t}(u_{n+1} - u_{n-1}) \tag{2.42}$$

Eq. (2.41),Eq. (2.42) are second order accurate approximations of the acceleration and velocity at time-step n, which can be seen by a Taylor Series expansion of the right hand sides:

$$\begin{aligned} \ddot{u}_n - (\frac{1}{(\Delta t)^2}(u_n + \Delta t\dot{u}_n + \frac{(\Delta t)^2}{2}\ddot{u}_n + \frac{(\Delta t)^3}{3}\frac{\partial^3 u_n}{\partial t^3} - \\ 2u_n + u_n - \Delta t\dot{u}_n + \frac{(\Delta t)^2}{2}\ddot{u}_n - \frac{(\Delta t)^3}{3}\frac{\partial^3 u_n}{\partial t^3} + O(\Delta t)^4)) = O((\Delta t)^2) \end{aligned} \tag{2.43}$$

$$\begin{aligned} \dot{u}_n - (\frac{1}{2(\Delta t)}(u_n + \Delta t\dot{u}_n + \frac{(\Delta t)^2}{2}\ddot{u}_n + \\ u_n - \Delta t\dot{u}_n - \frac{(\Delta t)^2}{2}\ddot{u}_n + O((\Delta t)^3)) = O((\Delta t)^2) \end{aligned} \tag{2.44}$$

The above two equations clearly demonstrate the second order accuracy of Eq. (2.18). In order to derive the stability region of Eq. (2.18) the linear elastic, damped, free vibration problem is solved:

$$\ddot{u}_n + 2\zeta\omega\dot{u}_n + \omega^2 u_n = 0 \tag{2.45}$$

The central difference approximation to this is obtained by plugging in, Eq. (2.41) and Eq. (2.42) and obtain:

$$(1+\zeta\omega\Delta t)u_{n+1}+((\Delta t)^2\omega^2-2)u_n+(1-\zeta\omega\Delta t)u_{n-1}=0 \tag{2.46}$$

Assuming a solution of the form $u_n=\lambda^n$:

$$(1+\zeta\omega\Delta t)\lambda^{n+1}+((\Delta t)^2\omega^2-2)\lambda^n+(1-\zeta\omega\Delta t)\lambda^{n-1}=0 \tag{2.47}$$

and upon dividing with λ^{n-1} the following arises:

$$(1+\zeta\omega\Delta t)\lambda^2+((\Delta t)^2\omega^2-2)\lambda+(1-\zeta\omega\Delta t)=0 \tag{2.48}$$

Conditional stability for Eq. (2.18) or Eq. (2.24), requires that the spectral radius $\rho(A)$ of the matrix A whose characteristic polynomial is Eq. (2.48) be $\rho(A)<1$. The case that the solution is two complex conjugate roots is:

$$\lambda_{1,2}=D\pm iE \tag{2.49}$$

with:

$$D=\frac{2-\omega^2(\Delta t)^2}{2(1+\zeta\omega\Delta t)},E=\frac{\omega\Delta t\sqrt{4-4\zeta^2-\omega^2(\Delta t)^2}}{2(1+\zeta\omega\Delta t)} \tag{2.50}$$

when

$$\omega\Delta t<\sqrt{4-4\zeta^2} \tag{2.51}$$

The case of two real roots is:

$$\lambda_{1,2}=D\pm E \tag{2.52}$$

with:

$$D=\frac{2-\omega^2(\Delta t)^2}{2(1+\zeta\omega\Delta t)},E=\frac{\omega\Delta t\sqrt{-4+4\zeta^2+\omega^2(\Delta t)^2}}{2(1+\zeta\omega\Delta t)} \tag{2.53}$$

when

$$\omega\Delta t\geq\sqrt{4-4\zeta^2} \tag{2.54}$$

The region of stability for the central difference method, can be obtained more easily if instead of analyzing in the μ-plane, the z-transform and Hurwitz matrices are used. Stability is determined by whether the characteristic equation has any non-negative real roots which is easy. The z-transform is :

$$\lambda = \frac{1+z}{1-z} \tag{2.55}$$

Using the above and substituting in Eq. (2.48):

$$(1+\zeta\omega\Delta t)(\frac{1+z}{1-z})^2 + ((\Delta t)^2\omega^2 - 2)\frac{1+z}{1-z} + (1-\zeta\omega\Delta t) = 0 \tag{2.56}$$

Multiplying by $(1-z)^2$ which is non-zero since $z < 1$ is required and simplifying:

$$(4-\omega^2(\Delta t)^2)z^2 + (4\zeta\omega\Delta t)z + (\Delta t)^2\omega^2 = 0 \tag{2.57}$$

The Hurwitz matrix of a polynomial equation of order p,

$$\sum_{i=0}^{p} c_i z^{p-i} = 0,\ c_0 > 0 \tag{2.58}$$

is given by:

$$H_{ij} = \begin{cases} c_{2j-i} & \text{if } 0 \le 2j-i \le p \\ 0 & \text{otherwise} \end{cases} \tag{2.59}$$

The real parts of the roots of Eq. (2.57) are negative if and only if the leading principal minors of the Hurwitz matrix are positive and c_0 is positive. The Hurwitz matrix for 2.57 is:

$$H = \begin{bmatrix} c_2 & 0 \\ c_0 & c_1 \end{bmatrix} \tag{2.60}$$

where $c_0 = 4-\omega^2(\Delta t)^2$, $c_1 = 4\zeta\omega\Delta t$ and $c_2 = (\Delta t)^2\omega^2$. The condition that the principal minors of the matrix are positive and $c_0 > 0$ yield the following set of equations:

$$\begin{aligned} 4-\omega(\Delta t)^2 &\ge 0 \\ 4\zeta\omega\Delta t &\ge 0 \\ (\Delta t)^2\omega^2 &\ge 0 \end{aligned} \tag{2.61}$$

From the requirements of Eq. (2.61) only the first one is nontrivial and leads to the following constraint that yields the critical time step:

$$\Delta t \leq min\left\{\frac{2}{\omega_i}\right\} \tag{2.62}$$

which is simply stated as:

$$\Delta t \leq \frac{2}{\omega_{max}} \tag{2.63}$$

This completes the stability analysis of Eq. (2.18).

Since the central difference formulation section requires mass proportional damping only to become diagonal, which limits the damping options, the analysis of an alternative difference scheme that allows for complete damping (mass and stiffness proportional) is presented. The basic assumption of the formulation of Eq. (2.24) lies in the alternative equilibrium equation:

$$M\ddot{u}_n + C\dot{u}_{n-1/2} + p_n^{int} = 0 \tag{2.64}$$

This assumption, of the damping forces lagging half a time step, is a first order accurate assumption since:

$$\begin{aligned} \dot{u}_n &= \dot{u}_{n-1/2} \\ \dot{u}_{n-1/2} &= \dot{u}_n + O(\Delta t) \end{aligned} \tag{2.65}$$

Eq. (2.24) is also analyzed using the method previously used for Eq. (2.18): Start with Eq. (2.64) for the free vibration of a linear elastic, damped system:

$$\ddot{u}_n + 2\zeta\omega\dot{u}_{n-1/2} + \omega^2 u_n = 0 \tag{2.66}$$

Substituting in the above Eq. (2.41) and Eq. (2.23) results in:

$$u_{n+1} + (2\zeta\omega\Delta t + \omega^2(\Delta t)^2 - 2)u_n + (1 - 2\zeta\omega\Delta t)u_{n-1} = 0 \tag{2.67}$$

which upon substituting with $u_n = \lambda^n$ and dividing with λ^{n-1}:

$$\lambda^2 + (2\zeta\omega\Delta t + \omega^2(\Delta t)^2 - 2)\lambda + (1 - 2\zeta\omega\Delta t) = 0 \tag{2.68}$$

Applying the z-transform the set of equations that arise from $c_0 > 0$ and the requirement that the principal minors of the Hurwitz matrix being positive yield:

$$\begin{aligned} 4 - 4\omega\Delta t - \omega^2(\Delta t)^2 &\geq 0 \\ 2\zeta\omega\Delta t &\geq 0 \\ (\Delta t)^2\omega^2 &\geq 0 \end{aligned} \tag{2.69}$$

Again, from the first one of the above constraints the critical time step for Eq. (2.24) is:

$$\Delta t \leq min\left\{\frac{2}{\omega_i}(\sqrt{1+\zeta_i^2} - \zeta_i)\right\} \tag{2.70}$$

but it is sufficient to use:

$$\Delta t \leq \frac{2}{\omega_{max}}(\sqrt{1+\zeta^2} - \zeta) \tag{2.71}$$

This completes the stability analysis for Eq. (2.24).

For the Newmark based method the stability discussion is in Chapter 4.

It is desirable to avoid solution of the eigenvalue problem required to find the highest frequency mode. For practical considerations the CFL condition by Courant et al. (1967), is satisfied for the wave propagation which essentially means to satisfying the requirement of Eq. (2.41) for each element in the mesh. So if λ is the critical length to be represented, the total time for a wave to travel past a point is $t_w = \frac{\lambda}{c}$ where c is the wave speed. Assuming that n time steps are necessary to represent the travel of this wave:

$$\Delta t = \frac{t_w}{n} \tag{2.72}$$

and the effective length of the finite elements should be:

$$L_e = c\Delta t \tag{2.73}$$

The formulation of Eq. (2.24) is first order accurate with respect to the equilibrium equation. The local truncation error from using Eq. (2.64) instead of using Eq. (2.40) is

defined as:

$$\begin{aligned}\tau &= C(\dot{u}_n - \dot{u}_{n-1/2}) \\ &= C(\dot{u}_n - (\dot{u}_n - \frac{\Delta t}{2}\ddot{u}_n)) \\ &= C\frac{\Delta t}{2}\ddot{u}_n \text{ where } \Delta t = t_{n+1} - t_n\end{aligned} \tag{2.74}$$

This concludes the discussion on explicit time integration methods. Having a diagonal left hand side (LHS) comes at the cost of losing the second order accuracy but adds a great computational advantage.

2.3 Software architecture for simulation of soil subdomains

This section describes the development of a software for the analysis of seismic wave propagation through a domain of interest and creates the basis for the target goal which is the coupled analysis of soil structure interaction using the DRM. The emphasis is on efficiency, scalability and speed of the solution using parallel computation.

2.3.1 Specification of the application

A traditional approach is used for the implementation of the simulation capability. The problem solution is decomposed into several phases as shown in Fig. 2.4 as opposed to an end to end approach, where all the capabilities required for the complete tackling of the problem are part of the same software. Rather than developing an entire application with all its necessary components from the beginning, the choice is to use components of an existing simulation framework. The reason is two-fold. On one hand, the literature is rich in developed finite element codes and secondarily when using available components, reduces the development time and allows for the focus to be shifted to the components that need the most attention, assuming of course that the usable components are of high quality and performance. In this case the new components that need attention and are of interest are related to large scale explicit and later mixed explicit-implicit time integration

in parallel computers and specifically efficient and scalable numerical methods, algorithms and data structures related to this. The basis framework that is selected is OpenSees (McKenna and Fenves, 2004).

The problem solving pipeline is essentially defining the application and is outlined as:

- Physical modeling
- Partitioning (serial or parallel)
- Solution and output data generation
- Visualization
- Understanding of the results

In summary this section presents: the new finite element classes shown in Fig. 2.5, and the new analysis classes shown in Fig. 2.8. These are new, additional to the framework implementations, and optimized for the problem under consideration and for the target platforms which are Datastar at the San Diego SuperComputing Center (SDSC, 2006) and Abe at the U.S. National Center for Supercomputing Applications (NCSA, 2008). Use of platform optimal math libraries ESSL (IBM, 2006) and MKL (INTEL, 2008) was done, wherever appropriate.

2.3.2 Brief overview of OpenSees

For a description of the basic architecture of the code and the current implementation one can refer to McKenna and Fenves (2004). OpenSees is a simulation framework targeting the analysis of the performance of structural and geotechnical systems subjected to earthquakes. It is developed using the object oriented programming paradigm and it is written in C++. The advantages of the adoption of the object-oriented paradigm (OOP) for engineering software design have been demonstrated in Fenves (1990).

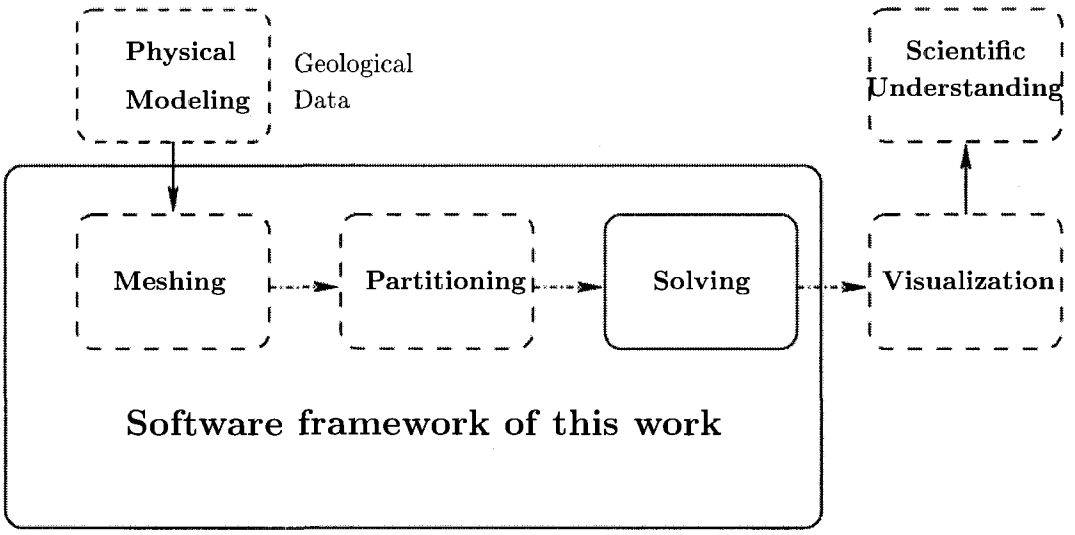

Figure 2.4. Traditional workflow pipeline

OpenSees contains modeling classes used to create the finite element model, finite element model classes that compose the finite element model and store the results, analysis classes that form and solve the equations and numerical classes that handle the numerical procedures used at the various solution classes. For an in detail description of the OpenSees architecture see McKenna (1997).

2.3.3 Finite element classes

2.3.3.1 Mesh3DSubdomain

The Mesh3DSubdomain class serves a dual role. Its first purpose is to create, partition, and prepare input for the main parallel program that will analyze a cubical three dimensional mesh using the DRM or some other loading pattern. It starts by the model details such as the number of elements in X,Y and Z directions, material properties and number of partitions of the domain's element graph. It performs a serial or parallel graph partition of the hexahedral element mesh using an interface to the METIS or ParMETIS Karypis (1998) library, and then proceeds in the generation of an input script, unique for

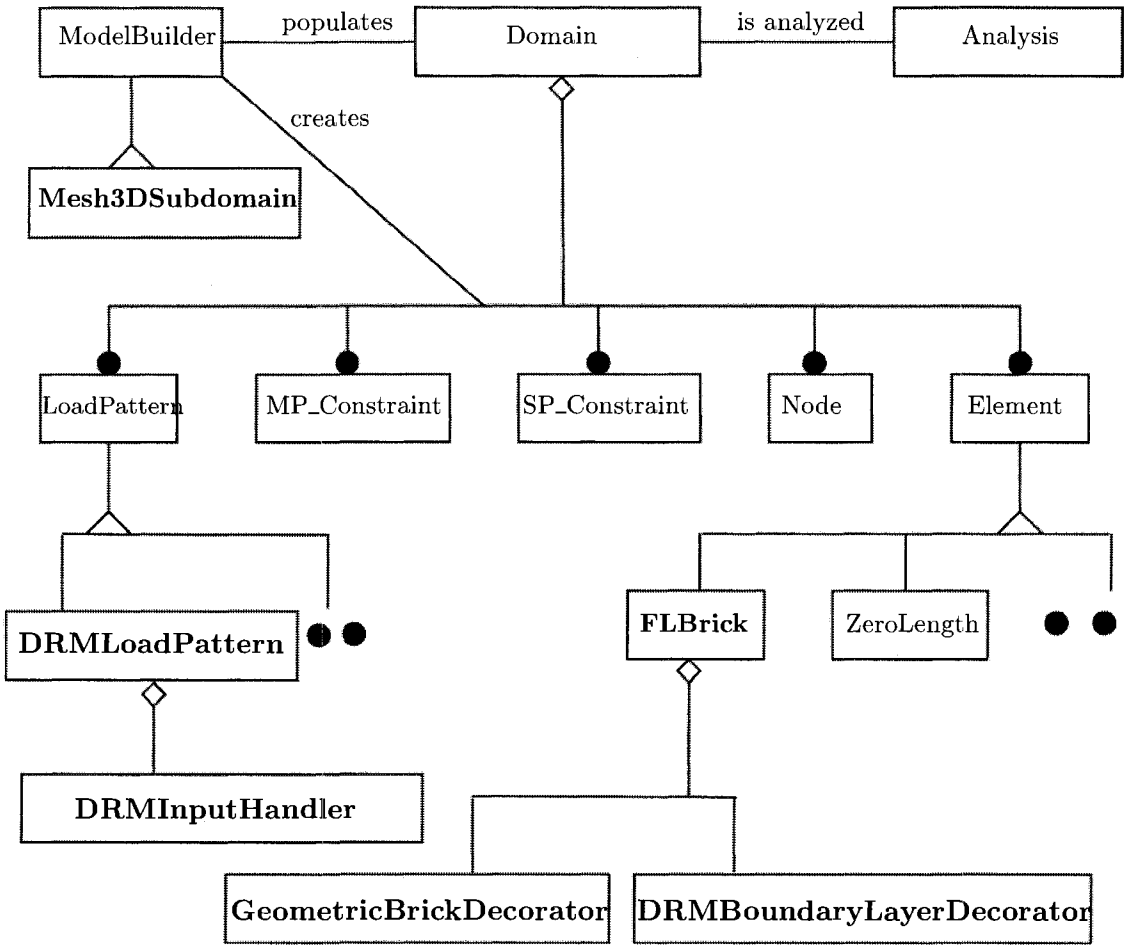

Figure 2.5. Modified OpenSees class diagram for the finite element method for soil subdomains (new implementations shown in bold)

each processor that will be participating in the solution of the problem. The resulting input script for each process, contains all the necessary data required for the corresponding partition to be created, such as node numbering, nodal coordinates, element connectivity, numbering and properties, for both the hexahedral and the absorbing boundary elements if present.

The second purpose of the Mesh3DSubdomain class is that inside the main parallel program it is used as the model builder class, therefore it is the entry point to the main program executed in each process. It reads the input data from the corresponding input file depending on the rank of the calling process in the MPI (Forum, 1994) communicator, and then constructs the necessary OpenSees objects that populate the Domain object that is created in each processor.

2.3.3.2 DRMLoadPattern class

The DRMLoadPattern class implements the basic functionality of any other LoadPattern instance in OpenSees, i.e. provides the **virtual void applyLoad(double time)** implementation that provides effective seismic loads according to the DRM. The method implementation is shown in Fig. 2.6. The helper classes that are related to this class are described further down.

2.3.3.3 PlaneDRMInputHandler

The PlaneDRMInputHandler class is a subclass of the DRMInputHandler class which is responsible for handling the input data required for the application of the Domain Reduction Method. As the name suggests it implements the case in which the boundaries Γ and Γ_e in Fig. 2.1, of the DRM box are planes. The input data of the DRM load pattern, which can be displacements, velocities, accelerations or combination of those, is provided in input files that contain the time histories for all the nodes of a mesh that is topologically equivalent to the e and b nodes of the mesh analyzed. This class is responsible for opening

```
void
DRMLoadPattern::applyLoad(double time)
{
  DRMBoundaryLayerDecorator *myDecorator = new DRMBoundaryLayerDecorator();
  myDecorator->setDomain(this->getDomain());
  Vector U(24);
  Vector Ud(24);
  Vector Udd(24);
  Vector load(24);
  U.Zero();
  Ud.Zero();
  Udd.Zero();
  load.Zero();
  myDecorator->setMap(this->eNodes);
  for(std::map<int,Element*>::iterator pos=this->elem.begin();
      pos!=this->elem.end(); pos++) {
     Element* ele = (Element*) pos->second;
     if (ele != 0) {
      U.Zero();
      Ud.Zero();
      Udd.Zero();
      load.Zero();
      myDecorator->setBrick(ele);
      this->myHandler->getMotions(ele, time, U, Ud, Udd);
      myDecorator->applyDRMLoad(load, U, Ud, Udd);
    }
  }
  delete myDecorator;
}
```

Figure 2.6. Implementation of **virtual void applyLoad(double time)** of DRMLoad-Pattern class

files and handling the request of an FLBrick element pointer of the Domain object that requests the necessary input wave field at its exact nodal locations at a specific instance of time. This handler class returns to the calling FLBrick element pointer the input motions required to compute its equivalent effective seismic input load. This is done by buffering in advance a certain number of time steps from the input time histories, performing spatial and temporal interpolations as required, as well as numerical integration or differentiation. The main function call provided by this class is **void getMotions(...)** , which upon completion has populated accordingly three vectors: U, Ud, Udd which correspond to the displacement, velocity and acceleration wave field at the nodes of the element. These are then used by the DRMLoadPattern class to apply the load for the specific decorated by the DRMBoundaryLayer decorator element pointer. The public members of the class are shown in the appendix A of the present document.

2.3.3.4 FLBrick

The FLBrick class is an implementation of a standard, three dimensional, isoparametric, linear elastic, brick element. All the element computations are optimized for the linear elastic case.

- The element stiffness matrix is computed once and stored in a static variable for all the elements residing in one processor and if needed is appropriately scaled with respect to the volume element and the Young's modulus of the material layer that it belongs to.
- The element mass matrix is computed once and stored in a static variable and if needed is scaled appropriately with respect to the density of the layer it belongs to.
- The element state determination is performed simply as $\mathbf{K}_e\mathbf{u}$ without going through the calculation of stresses via strains for speedup of the computation.
- The element needs to hold as a history variable for the time integration of the DRM

input, the displacements as computed from the previous time entry where DRM input is given. Previous time entry implies the time value t_i such that the DRM input is given, $t_i \leq t$ where t is the time for which the loading pattern requests the application of DRM effective seismic input and also $|t - t_i|$ is min.

2.3.3.5 GeometricBrickDecorator and DRMBoundaryLayerDecorator

The purpose of a decorator class is to dynamically attach additional functionality to an object (Gamma et al., 1995). It provides an alternative to subclassing for extending the functionality of an object. Decorators are also known as wrappers. The two decorator classes under discussion serve the purpose of adding necessary functionality to some objects of the FLBrick class, for the computations necessary for the construction and application of the effective seismic input via the DRM.

The GeometricBrickDecorator class implements all the operations needed for an element to be able to identify itself as to whether it is participating to the DRM loading pattern or not which is decided exclusively based on the element's geometrical information. With reference to Fig. 2.2 the single element layer Γ_e-Γ is essentially a special volume within the domain's volume. This volume needs be functionally identifiable in order for a specific element's volume to be able to figure out whether it belongs to it or not. In the most general case this is done by describing this volume by a specific point set such that a "point in volume" query suffices for all the elements of the domain in order to partition them in those belonging to this interface and those who don't. In a less geometrically general case this volume can be described by a set of curves that solely intersect the volumes of the participating elements. Finally the simplest case is when a certain number of planes suffices to describe this interface. Such is the case in a uniform hexahedral mesh, which also is the case of interest for the scope of this work. Specifically for the later case, for the purpose of the identification of the elements that participate in the DRM load pattern six coordinates are provided, min and max for the three directions and these are

the bounding coordinates of the cube whose exterior one-element layer comprises the set of elements that participate in the load pattern. Given those bounding coordinates and since the elements have access to the nodal coordinates of their node objects, via a simple linear traversal of the Domain object's elements and simple geometric comparisons, a data container is populated with the appropriate element pointers as part of the initialization of the PlanelDRMInputHandler instance in each processor. The GeometricDecorator class contains implementations of the necessary queries to handle all the possible geometric definitions of the layer Γ_e-Γ even though for the purpose of the present work the geometry is a simple cube. This class is also responsible for determining which of the element's nodes (assuming it belongs to the interface layer) are e and which are b nodes with respect to Fig. 2.2; this operation has similarities with the identification of the elements that participate in the loading pattern. In brief this decorator is performing geometric actions necessary for both the initialization and application of the DRM loading pattern.

The DRMBoundaryLayerDecorator class implements all the functionality required by an element that belongs to the enclosure of the two boundaries Γ_e and Γ in Fig. 2.2, in order for it to construct and apply its equivalent effective seismic input load according to Eq. (2.9). The DRMLoadPattern class makes a call to an appropriate element pointer which upon being decorated with this class, is executing the **void applyDRMLoad(...)**, which is passed as arguments the wave field at the nodal locations. The method's implementation is shown in Fig. 2.7.

2.3.4 Domain decomposition

The term refers to the method of solving boundary value problems (BVPs) by partitioning them into smaller independent BVPs and solving those on subdomains and then coordinating the solution among the subdomains. Consider a domain Ω split into element-wise disjoint sundomains Ω_1 and Ω_2 with the interface surface between the two subdomains $\Gamma_{1,2}$ as shown in Fig. 2.9. The unknowns of Eq. (2.24) are partitioned into three sets: $\mathbf{x}_1$

```
void DRMBoundaryLayerDecorator::applyDRMLoad(Vector &drmLoad,
                                             const Vector &displ,
                                             const Vector &veloc,
                                             const Vector &accel)
{
  Node *theNode;
  drmLoad.Zero();
  Vector load(3);
  this->computeDRMLoad(drmLoad, displ, veloc, accel);
  Node** nodes  = this->myBrick->getNodePtrs();
  for (int i=0; i<8; i++) {
    theNode = nodes[i];
    load.Zero();
    load(0) = drmLoad(i*3);
    load(1) = drmLoad(i*3+1);
    load(2) = drmLoad(i*3+2);
    theNode->addUnbalancedLoad(load);
  }
}
```

Figure 2.7. Implementation **void getMotions(...)** of PlaneDRMInputHandler class

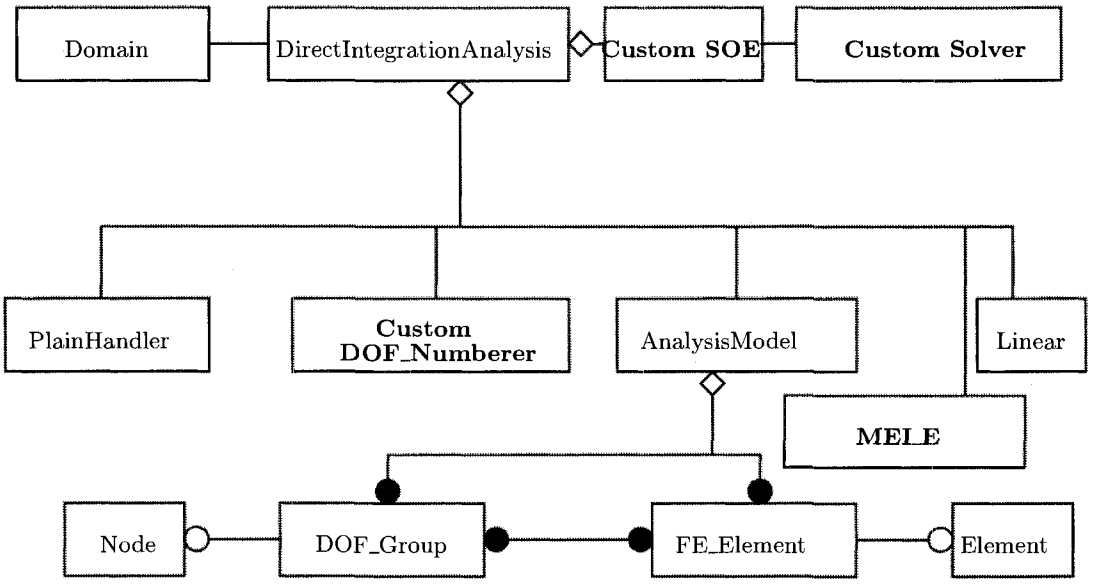

Figure 2.8. Modified OpenSees class diagram for transient analysis (new implementations shown in bold)

corresponding to Ω_1, those corresponding to Ω_2 denoted as $\mathbf{x}_2$ and those corresponding to $\Gamma_{1,2}$ denoted as $\mathbf{x}_3$.

In the general case the matrix equation is rewritten in block form as

$$\begin{bmatrix} \mathbf{A}_{11} & 0 & \mathbf{A}_{13} \\ 0 & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{bmatrix} \begin{Bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{Bmatrix} = \begin{Bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{Bmatrix} \tag{2.75}$$

where $\mathbf{A}_{33} = \mathbf{A}_{33}^1 + \mathbf{A}_{33}^2$, which means $\mathbf{A}_{33}$ contains both contributions from elements in Ω_1 and Ω_2. The unknowns $\mathbf{x}_1$ and $\mathbf{x}_2$ are easily eliminated from the system using the Schur compliment of $\mathbf{A}_{33}$ in $\mathbf{A}$. The resulting system yields

$$\mathbf{A}_{33}^S \mathbf{x}_3 = \mathbf{b}_3^S \tag{2.76}$$

where

$$\begin{aligned} \mathbf{A}_{33}^S &= \mathbf{A}_{33}^1 - \mathbf{A}_{31}\mathbf{A}_{11}^{-1}\mathbf{A}_{13} + \mathbf{A}_{33}^2 - \mathbf{A}_{32}\mathbf{A}_{22}^{-1}\mathbf{A}_{23} \\ &= \mathbf{A}_{33}^{1S} + \mathbf{A}_{33}^{2S} \end{aligned} \tag{2.77}$$

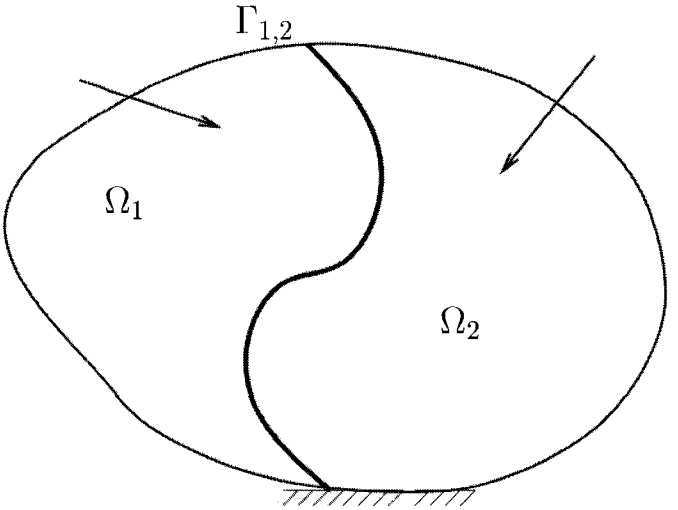

Figure 2.9. Domain Ω partitioned into two subdomains Ω_1 and Ω_2

and

$$\begin{aligned} \mathbf{b}_3^S &= \mathbf{b}_3^1 - \mathbf{A}_{31}\mathbf{A}_{11}^{-1}\mathbf{b}_1 + \mathbf{b}_3^2 - \mathbf{A}_{32}\mathbf{A}_{22}^{-1}\mathbf{b}_2 \\ &= \mathbf{b}_3^{1S} + \mathbf{b}_3^{2S} \end{aligned} \tag{2.78}$$

Upon determining the $\mathbf{x}_3$ the remaining unknowns of the problem are easily determined using Eq. (2.75). This concept can be easily extended to n subdomains. For a more in depth discussion of domain decomposition algorithms and methods refer to Toselli and Widlund (2005).

For the purpose of the parallel solution of the system of equations that arise from the explicit time integration of Eq. (2.8), a non-overlapping domain decomposition method is used which means that only interfaces of partitions are duplicated in the distributed matrix layout. The system solution does not require forming the LU factorization of the matrix because the left hand side is a diagonal matrix as it is shown in Eq. (2.17). The system of equations for the two subdomain case in blocked form is

$$\begin{bmatrix} \mathbf{A}_1 & 0 \\ 0 & \mathbf{A}_2 \end{bmatrix} \begin{Bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{Bmatrix} = \begin{Bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{Bmatrix} \tag{2.79}$$

where[1] $\mathbf{A}_i = \mathbf{A}_i^1 + \mathbf{A}_i^2$, which means that the diagonal block $\mathbf{A}_i$ contains both contributions from elements in Ω_1 and Ω_2. Similarly $\mathbf{b}_i = \mathbf{b}_i^1 + \mathbf{b}_i^2$, which means that the vector $\mathbf{b}_i$ contains both contributions from elements in Ω_1 and Ω_2. The diagonal matrix $\mathbf{A}_i$ gets assembled once in the first time step and is stored in process i of the MPI communicator., whereas the vector $\mathbf{b}_i$ need be assembled at each time step in process i with contributions of all the processes that process i shares an interface with. This is in essence what the customSOE and customSolver classes are solving in the most efficient possible way.

2.3.5 Parallel analysis in OpenSees

To utilize parallel computing, OpenSees implements the master-slave model, using an actor-shadow model, with process 0 (PE 0) being the master process and the rest of the processes in the MPI communicator being the slaves. The software design is shown in Fig. 2.10. A PartitionedDomain object is created in PE 0, and the element graph gets partitioned and each partition is broadcast to the corresponding slave process. In PE 0 no elements are stored but all the interface nodes are stored. Thus PE 0 stores the union of the "in pairs" intersections of the nodal graphs that get stored in all the other processes. Using a domain decomposition method (overlapping or non-overlapping), the interface is solved first in PE 0, then the result is broadcast to the other processes and the solution of each process's dofs takes place locally in its space. The communication between PE 0 and the rest of the processes is done via the actor-shadow model. In PE 0, besides the PartitionedDomain object, there exist also ShadowSubdomain objects one per process PE i of the communicator except for PE 0. In each process PE i an ActorSubdomain object is constructed, which communicates with its corresponding ShadowSubdomain object in PE 0. The PartitionedSubdomain solves the interface dofs then communicates them to the ActorSubdomain objects. This is done with a message that gets sent from PE 0 to PE i through the Channel that connects the ShadowSubdomain corresponding to PE i that

[1] subscript i indicates part of matrix owned by PE i

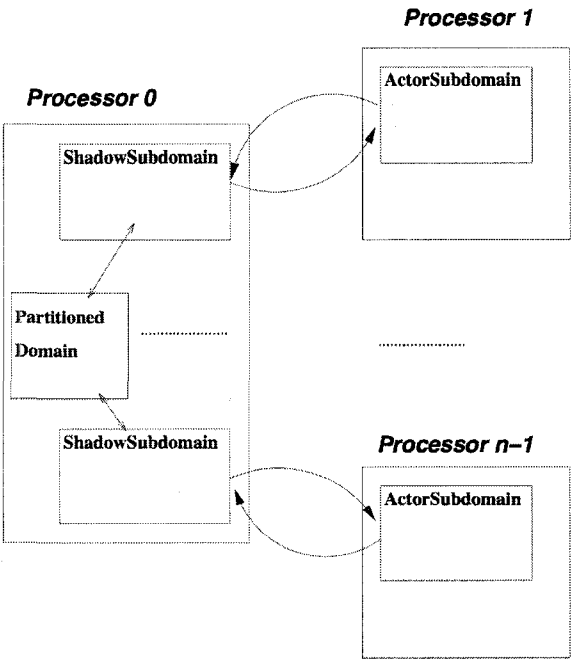

Figure 2.10. Parallel analysis architecture in OpenSees

resides in the address space of PE 0 to the ActorSubdomain that resides in the address space of PE i.

2.3.6 Parallel analysis implementation in the soil subdomain application

For parallel computing and to solve the governing equations for a model, one independent Domain object, is constructed in each process PE i of the communicator. Each Domain solves the total equation for each dof that it holds including the shared ones. The Domain objects do not need to communicate at all during the computation. This approach is in a way a small deviation from the general OpenSees architecture where a Domain object is considered a complete model rather than a partitioned component of one. Since the goal is a new implementation, this becomes acceptable. Each Domain object contains its own DirectIntegrationAnalysis object, which in turn contains a customSOE and a customSolver class instance. It is the responsibility of the customSOE and cus-

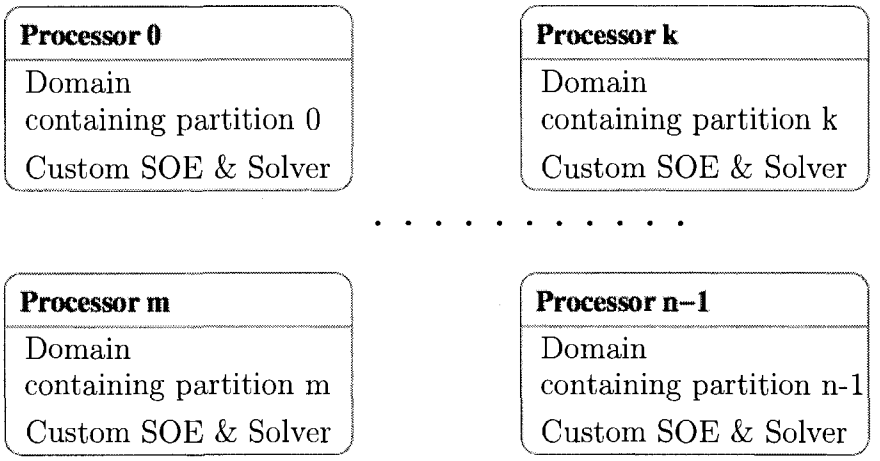

Figure 2.11. Parallel analysis architecture for simulation of sub-region

tomSolver objects instances contained in the Analysis object to identify and communicate with the appropriate processes for them to solve the equations for each dof they hold. This communication is done via MPI calls rather than using a variation of the OpenSees Channel object because it reduces the number of required function calls and since only distributed memory parallel computing is used. The design is shown in Fig. 2.11.

2.3.7 Analysis classes

2.3.7.1 CustomDOFNumberer

The numbering of the DOF, is the assignment of unique identification numbers to each DOF. The implementation of a custom degree of freedom numberer class is essential for speedup and performance. In a general distributed finite element analysis, it is necessary to number consistently a partitioned graph with unknown total number of edges and vertices, unknown and non constant in/out degree of its vertices. In this case, the parallel DOF graphs that are created, are just a set of vertices for an explicit solver, since there is no coupling in the equations and the set of edges is empty. While this does not affect the generality of the numberer, it reduces the amount of memory required for the storage of the

```
NUMBER-DOF
 1  ▷ executed by all ranks in parallel
 2  max ← 0
 3  for each vertex u in my_dof_group graph
 4        do if u.dofptr.dof_id.Size() ≥ max
 5              max ← u.dofptr.dof_id.Size()
 6  all_gather max, tmp_array
 7  max ← max[tmp_array]
 8  for each vertex u in my_dof_group graph
 9        do ptr ← u.dofptr
10              tmpsz ← ptr.dof_id.Size()
11              tag ← ptr.getTag()
12              for j ← 0 to (tmpsz − 1)
13                    do ptr.setDofID(j, (tag − 1) · max + j)
```

Figure 2.12. Parallel DOF numbering method

involved graphs. Within the OpenSees framework the DOF are grouped in DOF_Group objects that correspond to mesh nodes. An efficient algorithm, resulting in a nearly embarrassingly parallel way of performing this task is shown in Fig. 2.12. The key idea is that each process identifies the maximal number of degrees of freedom associated per DOF_Group, and upon a comparison among all the PEs, the global maximum is obtained on all processors. After this step each PE proceeds independent of the others. Upon the completion of the numbering phase, each domain object is populated with an array of integers containing the unique id for each DOF.

2.3.7.2 CustomSOE

The customSOE begins by implementing the **int setSize(Graph& theGraph)** method which identifies the size of system of equations that the processor is solving, numbers the degrees of freedom, and sets up the communication graph required for the solution in parallel. For that each process needs to figure out the subset of processes from

the MPI communicator that it needs to communicate with. Each PE identifies the subset of PEs it needs to communicate with during the solution phase, by sending to all the others the set of DOF it owns. The receiving PE performs a set intersection between its DsOF and the received ones of the potential neighbor. Should the intersection be non-empty, it identifies the sending PE as a neighbor. The receiving PE identifies during the intersection operation the locations of the local arrays that need to be stored. These are used when the associated data is sent/received to/from the corresponding neighbor. The pseudocode for this procedure is shown in Fig. 2.13. Two implementations are provided: one using an all-to-all multicast (which is the primitive MPI_Allgather) and another using a one-to-all multicast (MPI_Broadcast primitive). Fig. 2.13 shows the former, which is the default. The latter is useful because the space complexity of the all-to-all multicast is $O(nm)$, where n is the total number of PE and m is the number of DOF per PE. For problems with large sets of DOFs and limited memory, a one step at a time broadcast implementation reduces the space complexity to about twice the size of the local DOFs $O(m)$ and this is the scalable version of the algorithm since it has a fixed memory requirement independent of the size of the communicator.

That the set of DOFs of each process is sorted using the quicksort algorithm that takes $O(nlog(n))$ time and is considered ideal for sorting arrays in place. Quicksort is a recursive divide-and-conquer algorithm and in practice is the fastest known comparison-based sort method for arrays even though it has worst-case running time $\Theta(n^2)$. When properly implemented it does run though in $O(nlogn)$. This sorting allows for $O(logn)$ search time in the data structures that contain the dofs and also $O(n+m)$ intersection time between two sets of dofs. For algorithms and analysis of their running time the reader is referred to the standard reference Cormen et al. (2001).

Each process assembles the total vector of its shared dofs that it shares with the entire list of vertices in its edge list, and broadcasts it to them. These mappings are stored at each process to allow for minimal size of required communicated messages. Since solution

is done using explicit integration in time and the left hand side is mass dependent and constant Eq. (2.17), at each time step only the right hand side needs assembly Eq. (2.30). The left hand side assembly is performed once at the first time step and no further communication takes place for it. Aside from the previously mentioned characteristics, the customSOE class provides the exact same method calls that any SOE class is required to implement within the OpenSees architecture customized for an explicit, diagonal system of equations.

2.3.7.3 CustomSolver

The purpose of the customSolver class is to solve the Eq. (2.31). At each time step it loops through the edge list of the process (communication graph) to assemble the right hand side, and after assembly it solves the equations for the total dofs. The customSolver class provides the required method implementations customized for an explicit system of equations and parallel processing using MPI. The pseudocode for the solution phase is given in Fig. 2.14. The key steps are the local computations on each PE, where according to the integration algorithm the **A** and **b** components are assembled. Using the cached sizes that correspond to the per PE communication requirements the send and receive buffers are allocated. This operation is only performed once. Each rank populates the send buffers that correspond to each neighbor PE with the required subset of the local **b**. **A**, depending on the problem and algorithm, is only communicated at the first step if it is fixed, otherwise on each time increment. The generic communication algorithm is shown in Fig. 2.15. The placeholders **comm_op_1** and **comm_op_2** can each call either MPI_Irecv or MPI_Isend and MPI_Send or MPI_Isend or MPI_Recv or MPI_Irecv correspondingly. Clearly **comm_op_1** can only be non-blocking. An implementation for a fully non-blocking version with receives posted first is shown in Fig. 2.16. Part of the local computation is interleaved with the communication calls. The benefit is minimal for linear problems with explicit time integration in homogeneous clusters, as the work of Danielson

```
SET-UP-COMMUNICATION-GRAPH(my_dofs_array)
 1  ▷ executed by all ranks in parallel
 2  List adj, adj_sizes, adj_arrays
 3  quicksort(my_dofs_array)
 4  local_size ← length[my_dofs_array]
 5  tmp_array[size_of_comm]
 6  all_gather local_size, tmp_array
 7  max_size ← max[tmp_array]
 8  num_neighbors ← 0
 9  recv_buf[sum of sizes]
10  allgather my_dofs, max_size, recv_buf, tmp_array
11  for j ← 0 to (size_of_comm −1)
12        do
13           ▷ Intersect my local dofs with each received
14           ▷ dofs to figure out possible neighbors,
15           ▷ the size of the intersection and the
16           ▷ locations in my array
17           if my_rank ! = j
18              then tmp_dofs ← recv_buf[j · max_size]
19                   tmp ← intersect(my_dofs, tmp_dofs)
20                   if tmp empty continue
21                   adj[num_neighbors] ← j
22                   adj_sizes[num_neighbors] ← length[tmp]
23                   adj_arrays[num_neighbors] ← tmp
24                   num_neighbors ← num_neighbors +1
25  ▷ Finally perform local renumbering of dofs
26  ▷ according to the sorted locations
27  ▷ (preserves location in local arrays)
```

Figure 2.13. Algorithm for setting up the communication graph

and Namburu (1998) demonstrated, but it is still recommended. The decision of the realization of the placeholders **comm_op_1** and **comm_op_2**, depends on the platform, problem and hence message sizes. This is done as is part of the tuning of the code prior to porting to a new machine.

2.3.8 Comparison of the two architectures

An example that illustrates the difference between the two architectures is presented in this section. Consider a 2D domain partitioned into 64 partitions as shown in Fig. 2.17. The dashed lines represent the limits of the partitions, thus refer to shared degrees of freedom.

The OpenSees architecture communication graph for the analysis of this problem is shown in Fig. 2.18. An additional process used, i.e. a total of 65 processes, for the role of the master process. Alternatively one could use 63 partitions of the initial domain and one for the role of the master process. Even if a partition were allowed on PE 0 too, that would create immediately a large imbalance since PE 0 is responsible for both coordination of the entire analysis and local computation. For the purpose of having equally size partitions in both codes 64 processes plus one master process where used for OpenSees. The communication graph for this architecture is shown in Fig. 2.19.

Graph theory terminology is introduced to facilitate the comparison. A graph is a set of vertices connected by edges. For the purposes of this work all edges are bidirectional, which means that reference is made to undirected graphs only, since they represent communication between two processes. Graphs can be represented via matrices or lists. Here the matrix representation is used. Adjacency matrix $A = a(i,j)$ of an undirected graph $G = (V, E)$ having $|E|$ edges and $|V|$ vertices is a $|V| \times |V|$ matrix such that:

$$A(i,j) = A(j,i) = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{2.80}$$

```
DIAGONAL-SOLVE
 1  ▷ Set according to time integration algorithm
 2  myA ,myB, myX
 3  List adj,adj_sizes, adj_arrays
 4  ▷ Send and Recv buffers
 5  List adj_As, adj_Bs, adj_Ar, adj_Br
 6  ▷ Each rank populates its send buffers
 7  for j ← 0 to (length[my_neighbors] − 1)
 8        do tmp_size ← length[adj_arrays[j]]
 9           if A_not_set
10              then for i ← 0 to (tmp_size −1)
11                         do tmp ← adj_arrays[j][i]
12                            adj_As[j][i] ← myA[tmp]
13                            adj_Bs[j][i] ← myB[tmp]
14              else  for i ← 0 to (tmp_size −1)
15                         do tmp ← adj_arrays[j][i]
16                            adj_Bs[j][i] ← myB[tmp]
17 ▷ Perform the communication steps according to Fig. 2.15
18 for j ← 0 to (length[my_neighbors] − 1)
19       do tmp_size ← length[adj_arrays[j]]
20          if A_not_set
21             then for i ← 0 to (tmp_size −1)
22                         do tmp ← adj_arrays[j][i]
23                            myA[tmp] + ← adj_Ar[j][i]
24                            myB[tmp] + ← adj_Br[j][i]
25             else  for i ← 0 to (tmp_size −1)
26                         do tmp ← adj_arrays[j][i]
27                            myB[tmp] + ← adj_Br[j][i]
28 ▷ If A is constant during all the steps
29 for j ← 0 to (local_size −1)
30       do myA[j] ← 1/myA[j]
31 A_not_set ← false
32 ▷ Solve
33 for j ← 0 to (local_size −1)
34       do myX[j] ← myA[j] · myB[j]
```

Figure 2.14. Algorithm for solution phase

```
COMMUNICATION
1  for j ← 0 to (length[my_neighbors] − 1)
2        do comm_op_type_1 j, adj_sizes[j], buf_type[j]
3  for j ← 0 to (length[my_neighbors] − 1)
4        do comm_op_type_2 j, adj_sizes[j], buf_type[j]
5  Wait_All
```

Figure 2.15. Generalized communication structure

```
NON-BLOCKING
 1  if A_not_set
 2     then for j ← 0 to (length[my_neighbors] − 1)
 3                do Irecv j, adj_sizes[j], adj_Ar[j]
 4                   Irecv j, adj_sizes[j], adj_Br[j]
 5          for j ← 0 to (length[my_neighbors] − 1)
 6                do Isend j, adj_sizes[j], adj_As[j]
 7                   Isend j, adj_sizes[j], adj_Bs[j]
 8     else  for j ← 0 to (length[my_neighbors] − 1)
 9                do Irecv j, adj_sizes[j], adj_Br[j]
10          for j ← 0 to (length[my_neighbors] − 1)
11                do Isend j, adj_sizes[j], adj_Bs[j]
12  ▷ Overlap Computation by solving on all non-shared dofs
13  for j ← 0 to (local_size −local_shared)
14       do myX[j] ← myA[j] · myB[j]
15  Wait_All
```

Figure 2.16. Non blocking communication with receives first

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

Figure 2.17. Partitioned 2D domain

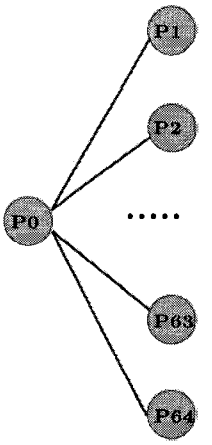

Figure 2.18. Resulting communication graph for OpenSees

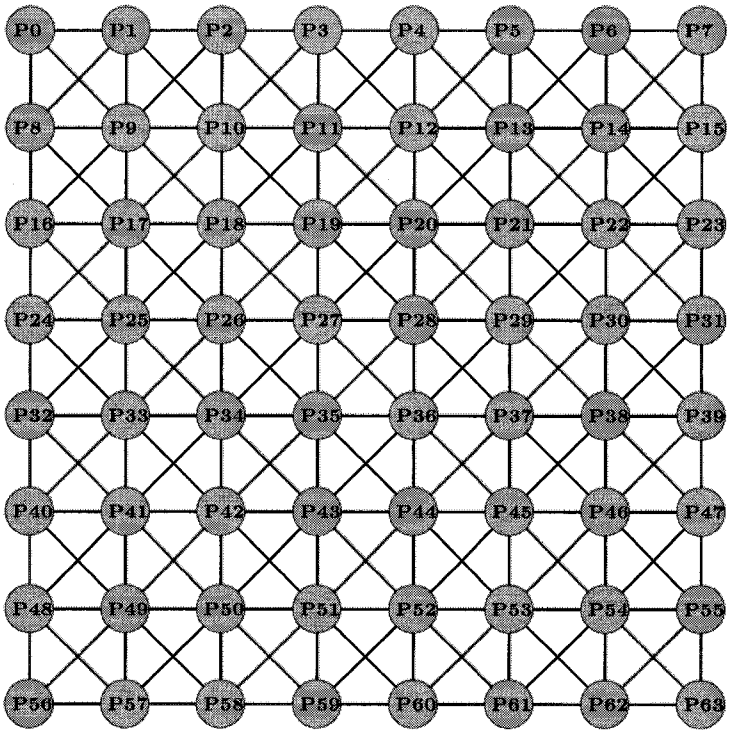

Figure 2.19. Resulting communication graph for this application

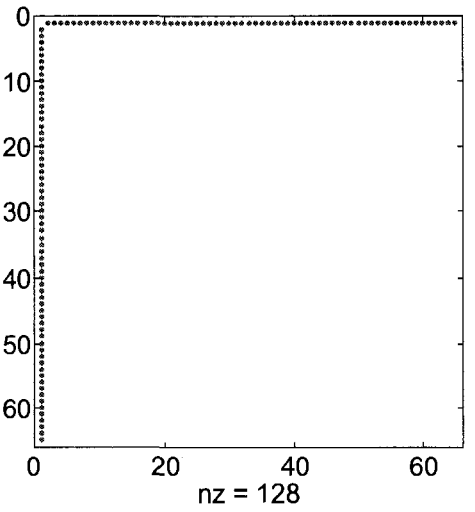

Figure 2.20. Sparsity of adjacency matrix for OpenSees, dots representing non-zero entries

The adjacency matrix representing the graph shown in Fig. 2.18 describing communication during parallel analysis in OpenSees for the domain shown in Fig. 2.17 is shown in Fig. 2.20. The adjacency matrix representing the graph shown in Fig. 2.19 describing communication during parallel analysis in this application for the domain shown in Fig. 2.17 is shown in Fig. 2.21.

The general OpenSees method of parallel processing does not allow for scalability to a large number of processors, because of four bottlenecks in the pipeline. First, the size of the total PartitionedDomain object is limited to the memory capacity of process PE 0. Second, the number of total interface dofs, is also limited by the memory capacity of process PE 0. Third, while PE 0 solves the interface problem all the other processes remain idle waiting for the broadcasting of the interface dofs. Lastly, the fact that the communication is taking place between PE 0 and all the other processes PE i creates a significant unbalance in the communication graph. Essentially the architecture in Fig. 2.18 for a communicator of size n requires a bus $n-1$ times as fast for node with rank 0, compared to the buses of the other nodes. In the architecture developed herein, the

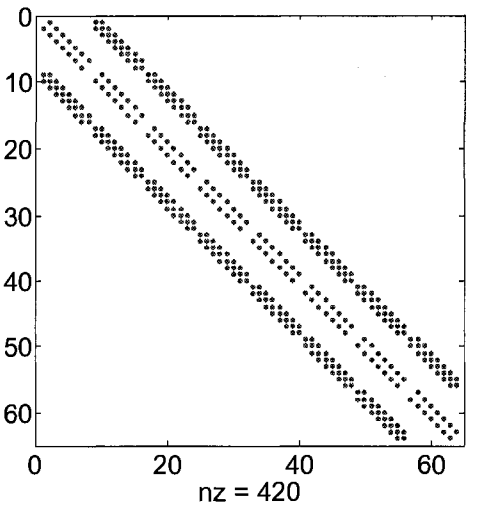

Figure 2.21. Sparsity of adjacency matrix for this application, dots representing non-zero entries

goal was to eliminate the bottlenecks and produce an application that can scale to very large numbers of processors. Note that the in/out degree of any vertex in Fig. 2.19 is significantly smaller than the resulting from the architecture of Fig. 2.10 where the degree of the vertex that corresponds to processor 0 is $n-1$, where n is the number of processes that participate in the communicator, and every other vertex has in/out degree of 1. For the application of Fig. 2.19 the max in/out degree is 8 or 26 for a regular 2 or 3 dimensional grid and this presents a great advantage. A communication graph such that of Fig. ?? is better balanced and therefore the application is very scalable.

2.4 Analysis of scalability and performance

The strong and weak scalability of the explicit finite element platform that was presented in the previous section, is studied. A benchmark problem with a simple loading is modeled and is used to assess the performance.

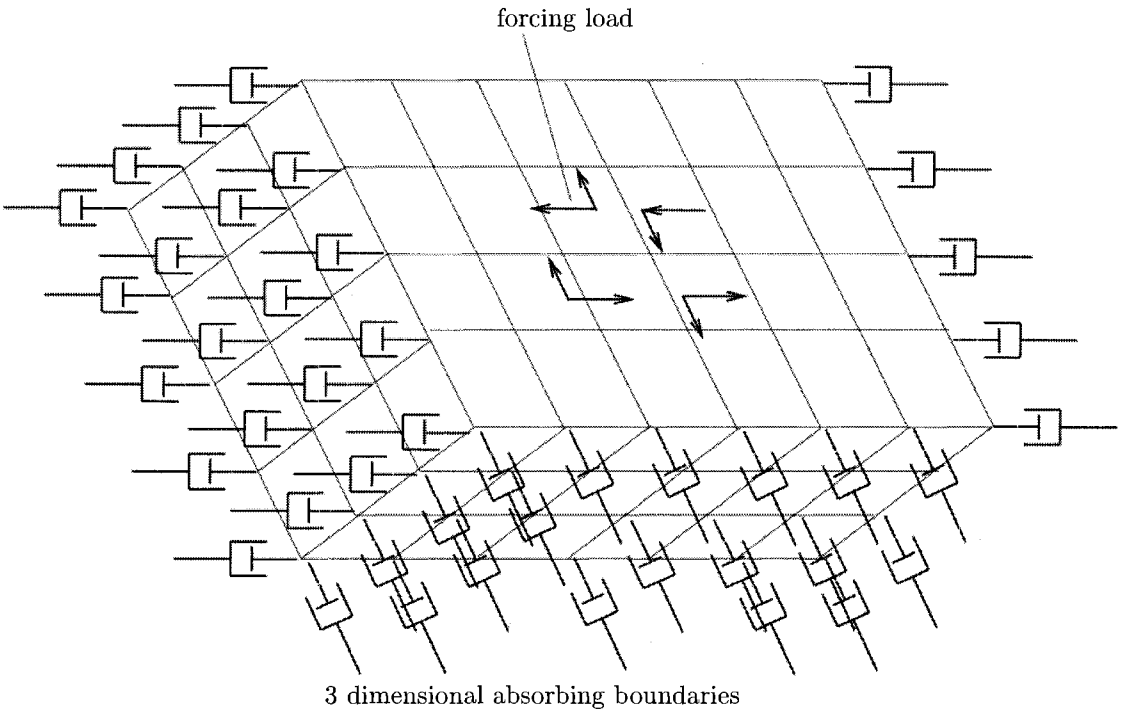

Figure 2.22. Three dimensional benchmark problem used for scalability measurements

2.4.1 Benchmark model description

A semi-infinite half-space is modeled using hexahedral isoparametric elements for the soil and absorbing boundaries to account for the energy absorption at infinity. Graphically the problem is shown in Fig. 2.22. The volume of the model is a 1400 m $\times$ 900m $\times$ 300 m region meshed using 20 m, 10 m, 5 m and 2.5 m cubical elements. In all cases the model was analyzed for 20000 time steps so that the initialization costs become amortized and also in order to have as good as possible estimates of the time per element-per time step-per PE metric. The loading for the testbed example was a simple forcing bi-couple in the center of the domain at the free surface. The details of each model that was analyzed are presented in Table 2.1.

Table 2.1. Benchmark run data

	element size (m)	# elements	# nodes	# DOFs	PE range
Run A	20	54026	59032	156768	2-32
Run B	10	404751	424512	1193283	16-256
Run C	5	3130301	3208822	9307563	128-2048
Run D	2.5	24615801	24928842	73515123	1024-2048

2.4.2 Strong and weak scalability analysis

Strong or fixed-size scalability and weak or isogranular scalability for this application is examined for the benchmark problem. Strong scalability refers to the case when the problem size is fixed and by increasing the number of processors used for parallel computation the goal is to proportionally decrease the time required for the solution. Strong scalability is measured using two standard metrics, the speedup and the parallel efficiency metric. Assume a fixed size problem analyzed with k PE's and then with n PE's, and also assume k$\leq$n. The speedup for the case of n PE's is defined as $S(n) = \frac{T(k)}{T(n)}$. For the same example the parallel efficiency is defined as $E(n) = \frac{kT(k)}{nT(n)}$.

Weak scalability refers to the fact that ideally and as long as the number of elements per processor remain constant the analysis time, regardless of the size of the model, will remain constant. That of course is impossible given that it implies a zero cost of communication among processing units as well as other idealizations. Weak scalability is shown via the time/element/step metric.

The detailed results for each run shown in Table 2.1, are shown in Table 2.2 through Table 2.5. Strong scalability is plotted in Fig. 2.23 and weak scalability is plotted in Fig. 2.24.

2.4.3 Analysis of the performance results

Scalability is achieved for all ranges of processing unit numbers, up to 2048 processors, almost linearly, as the speedup row shows in all the tabulated results. In some cases it

Table 2.2. Run A, detailed performance data

PEs	2	4	8	16	32
Elements/PE	23625	11812	5906	2953	1467
EToE time (sec)	11847.19	5569.78	3052.76	1414.22	846.85
speedup	1	2.127	3.881	8.377	13.99
relative speedup	1	2.127	1.82	2.16	1.67
parallel efficiency	1	1.06	0.97	1.04	0.87
comm time (% EToE time)	1.05	2.31	4.98	10.909	13.91
EToE time/elem/step (μsec)	25.1	23.6	25.8	23.9	28.9

Table 2.3. Run B, detailed performance data

PEs	16	32	64	128	256
Elements/PE	23625	11812	5906	2953	1467
EToE time (sec)	13621.23	7229.51	3188.34	1411.45	793.38
speedup	1	1.884	4.272	9.651	17.169
relative speedup	1	1.884	2.27	2.26	1.78
parallel efficiency	1	0.94	1.06	1.20	1.07
comm time (% EToE time)	6.83	5.79	12.33	12.26	20.38
EToE time/elem/step (μsec)	28.8	30.6	26.9	23.9	27

Table 2.4. Run C, detailed performance data

PEs	128	256	512	1024	2048
Elements/PE	23625	11812	5906	2953	1467
EToE time (sec)	15232.92	7616.11	3633.15	1762.89	981.90
speedup	1	2.0	4.19	8.64	15.52
relative speedup	1	2.0	2.10	2.06	1.79
parallel efficiency	1	1	1.04	1.08	0.97
comm time (% EToE time)	8.70	9.92	14.86	22.56	27.31
EToE time/elem/step (μsec)	32.2	32.2	30.8	29.8	33.4

Table 2.5. Run D, detailed performance data

PEs	1024	2048
Elements/PE	23625	11812
EToE time (sec)	15863.45	7996.29
speedup	1	1.98
relative speedup	1	1.98
parallel efficiency	1	0.99
comm time (% EToE time)	14.07	18.66
EToE time/elem/step (μsec)	33.6	33.8

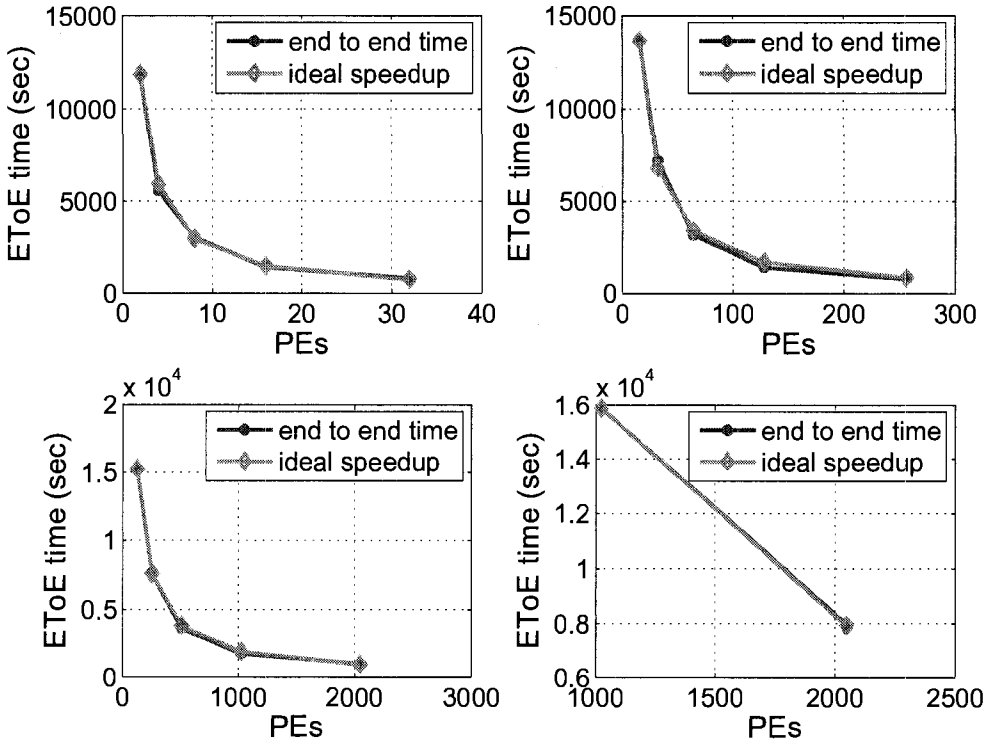

Figure 2.23. Fixed-size, scalability plot. Row 1 is runs A and B, Row 2 is runs C and D

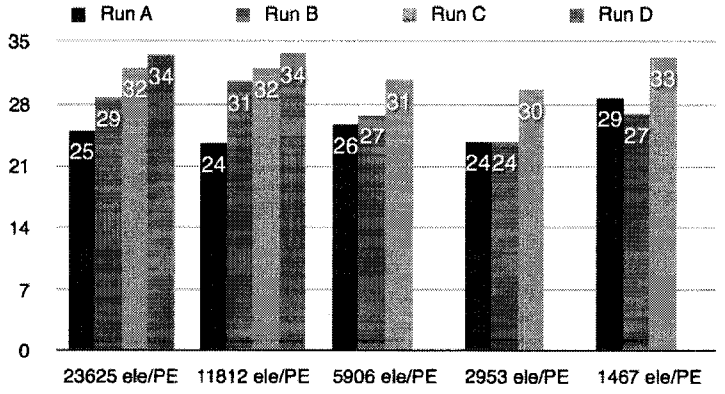

Figure 2.24. Isogranular scalability plot. Time per element per step (μsec)

appears to even be superlinear. This indicates that the selected problem sizes might be small enough sometimes, and thus fits the processor's cache. Yet even so, the relative speedup is very good also. Ideally it should always be 2 and is in the worse case, for the large problems 1.78. Communication time appears to be at the most 27% as shown in Table 2.4 which is considered satisfactory given that most finite element codes have about 25% communication cost.

The communication costs indicate that even for this simple mesh because the partitioning is only hexahedral element based, there exists an unbalance in the computational units among different processes of the same communicator. Even though processors have the same number of hexahedral elements some of them that contain near boundary regions also have absorbing boundary elements whose computational load results in this unbalance. This is also an issue for when analyzing a model using the DRM load pattern. The elements that participate in the DRM load pattern have a significantly higher computational load compared to the hexahedral elements and thus constitute an additional imbalance reason. This indicates that a dynamic load balancing strategy may be needed especially if nonlinear soils will be considered. This remains a future goal outside the scope of this work.

The computational costs can be split in two components; one is the element state determination, which consists of the time the finite element objects spend to compute their internal state, thus using the computed displacements from the previously converged state to evaluate the resisting forces for the right hand side of the current time step, and second the time spent for the solution of the equation of motion for the current time step. Ideally both of these steps need be balanced for a minimal communication. Since the model contains boundary elements of different type from the regular hexahedrals, that is not possible. A weighted partition of the element graph would succeed in balancing the state determination part of the computation resulting in a more unbalanced solution phase. The solution phase was found to be more important to maintain balanced and

thus the partitioning is based only on the hexahedrals. It is a partitioning that achieves minimal communication costs and leaves some unbalance in the computational loads and in particular in the element state determinations.

As for the weak scalability results, the time per element/step metric results are considered high. An improvement of about three times is desired. The case of 11812 elements per processor shows the best balance. This metric indicates that it is needed to perform core optimizations for the way the code allocates and handles memory for the objects and that some improvement in the data structures is needed. This is not considered within the scope of this work but is a task for the future.

2.4.4 Analysis of communication cost for the solver

In this section detailed results are presented for the communication costs of each run and how these are distributed among the main MPI calls that the code does. The results presented in Table 2.6 through Table 2.9 show that in the case of Run A, the communication time is mainly spent in the **MPI_Waitall(...)** function where as for Runs B,C,D the governing function is **MPI_Send(...)** and secondarily **MPI_Waitall(...)** .

Table 2.6. Run A, detailed communication time data

PEs	2	4	8	16	32
Elements/PE	23625	11812	5906	2953	1467
EToE time (sec)	11847.19	5569.78	3052.76	1414.22	846.85
comm time (% EToE)	1.05	2.31	4.98	10.909	13.91
MPI_Send (% EToE)	0.007	0.03	0.07	3.013	5.45
MPI_Irecv (% EToE)	0.0048	0.012	0.024	0.038	0.058
MPI_Waitall (% EToE)	1.032	2.248	4.818	7.8	8.37
MPI_Bcast (% EToE)	0.00014	0.00034	0.00057	0.013	0.003933
MPI_Barrier (% EToE)	0	0	0	0.000248	0.00011
MPI_Allgather (% EToE)	0.00499	0.024	0.0598	0.0469	0.026

Table 2.7. Run B, detailed communication time data

PEs	16	32	64	128	256
Elements/PE	23625	11812	5906	2953	1467
EToE time (sec)	13621.23	7229.51	3188.34	1411.45	793.38
comm time (% EToE)	6.83	5.79	12.33	12.26	20.38
MPI_Send (% EToE)	5.2083	4.31	8.313	9.036	12.044
MPI_Irecv (% EToE)	0.00966	0.01632	0.03714	0.04321	0.0738
MPI_Waitall (% EToE)	1.355	1.312	3.863	0.03117	8.175
MPI_Bcast (% EToE)	0.00864	0.001902	0.000455	0.03607	0.03607
MPI_Barrier (% EToE)	0.00001284	0.00005857	0.0000838	0.0135	0.01352
MPI_Allgather (% EToE)	0.26	0.152	0.11263	0.0472	0.0341

Table 2.8. Run C, detailed communication time data

PEs	128	256	512	1024	2048
Elements/PE	23625	11812	5906	2953	1467
EToE time (sec)	15232.92	7616.11	3633.15	1762.89	981.90
comm time (% EToE)	8.70	9.92	14.86	22.56	27.31
MPI_Send (% EToE)	6.69	7.925	10.63	16.80	18.851
MPI_Irecv (% EToE)	0.02973	0.03731	0.05238	0.067	0.09
MPI_Waitall (% EToE)	1.625	1.673	3.6835	5.384	7.1033
MPI_Bcast (% EToE)	0.00795	0.01922	0.050	0.1486	0.489
MPI_Barrier (% EToE)	0.00104	0.002945	0.0159	0.09349	0.074
MPI_Allgather (% EToE)	0.3380	0.2658	0.1528	0.06206	0.0408

Table 2.9. Run D, detailed communication time data

PEs	1024	2048
Elements/PE	23625	11812
EToE time (sec)	15863.45	7996.29
comm time (% EToE)	14.07	18.66
MPI_Send (% EToE)	10.28	13.86
MPI_Irecv (% EToE)	0.04486	0.05374
MPI_Waitall (% EToE)	3.172	3.81
MPI_Bcast (% EToE)	0.075	0.193
MPI_Barrier (% EToE)	0.02826	0.4675
MPI_Allgather (% EToE)	0.047	0.2722

Chapter 3

Analysis of soil subdomains for near-fault ground motion

This chapter presents a one dimensional wave propagation example to verify the correctness of the solver and the FE code in general. The results are compared against closed form solution. Next, a near fault region in Downtown Los Angeles, adjacent to the Puente Hills fault is analyzed, serving as the first application example. The analyses use low frequency and broadband input. Two soil types are considered for the low frequency input, the original one as well as a softer soil. The Domain Reduction Method (DRM) is applied in order to provide for the effective seismic input in the soil subdomain. The DRM method is validated for the soil profile used to generate the input motion.

3.1 A one dimensional wave propagation example

Suppose a one dimensional half-space of an elastic material that is initially undisturbed. That is $u(x,t) = 0$ for $t \leq 0$ and subjected to the normal-stress boundary

condition $\sigma_{11}(0,t) = p(t)$. We know from elasticity that

$$\sigma_{11} = (\lambda + 2\mu)\epsilon_{11} = (\lambda + 2\mu)\frac{\partial u}{\partial x} \tag{3.1}$$

This allows us to rewrite the boundary condition as

$$\frac{\partial u}{\partial x}(0,t) = \frac{p(t)}{\lambda + 2\mu} \tag{3.2}$$

Since we are solving in a half-space we can assume the solution to only contain a forward propagation wave $u(x,t) = f(\xi) = f(t - \frac{x}{c})$ where c is the wave propagation speed of the material. By simple differentiation and setting $x = 0$ for the boundary condition we obtain that

$$\frac{df}{dt} = -c\frac{p(t)}{\lambda + 2\mu} \tag{3.3}$$

and then solving for $f(\xi)$ we have

$$u(x,t) = f(\xi) = f(0) - \frac{c}{\lambda + 2\mu}\int_0^{t-\frac{x}{c}} p(\tau)d\tau \tag{3.4}$$

Now for the initial condition to verify, it follows that $f(0) = 0$ and thus finally we have for the solution

$$u(x,t) = -\frac{c}{\lambda + 2\mu}\int_0^{t-\frac{x}{c}} p(\tau)d\tau \tag{3.5}$$

A unit area bar of length 400 m with $V_S = 50$ m/sec and $V_P = 100$ m/sec is modeled with 400 FLBrick elements (Section 2.3.3.4), and solved numerically. We apply the normal stress as a forcing input in the four nodes of the first element and use absorbing boundary elements to account for the energy absorption at infinity. To simulate the one dimensional effect the degrees of freedom that are perpendicular to the axis of the model are fixed. The input is a sinusoidal force $F(0,t) = sin(2\pi f_0 t)$ ($\frac{1}{4}$ of $F(0,t)$ per node). The time integration is performed using all three schemes discussed in the previous chapter.

The comparison of the closed form solution from Eq. (3.5) and the numerical solution is shown in Fig. 3.1. The results present excellent agreement, and validate the solution method and parallel numerical procedure.

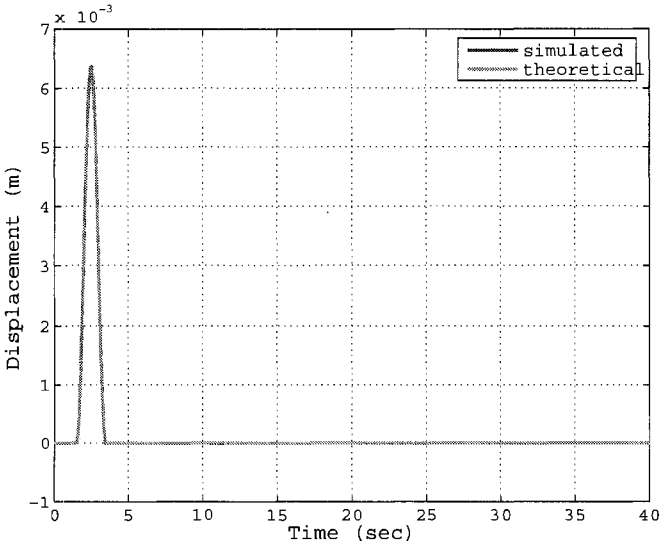

Figure 3.1. Comparison of closed form solution and numerical solution using our custom code for a simple one dimensional wave propagation problem

3.2 Analysis of soil subdomain near the Puente Hills fault

A near fault region of size 1000 m by 500 m by 100 m is analyzed for low frequency and broadband input using the original soil profile as well as a modified softer one. The lower left corner of the region or DRM box, has latitude 34.0922695 and longitude -118.328208. The DRM input was provided by a large scale simulation that was conducted under the supervision of Professor Jacobo Bielak of Carnegie Mellon University. The position of the region of interest for our problem with respect to the fault projection is shown in Fig. 3.2. The soil profile of the model is given in Table 3.1.

Table 3.1. Material properties for actual, simplified, layered soil profile

Layer	Density (t/m^3)	V_P (m/sec)	Q_P	V_S (m/sec)	Q_S	Thickness (m)
1	1.5	404.7	8.094	164.9	4.047	5.0
2	1.5	1163.5	23.27	385.4	11.63	10.0
3	1.5	1337.3	26.74	482.2	13.37	40.0
4	1.714	1622.4	243.36	584.3	121.68	50.0
5	2.054	2372.9	355.80	651.3	177.9	50.0

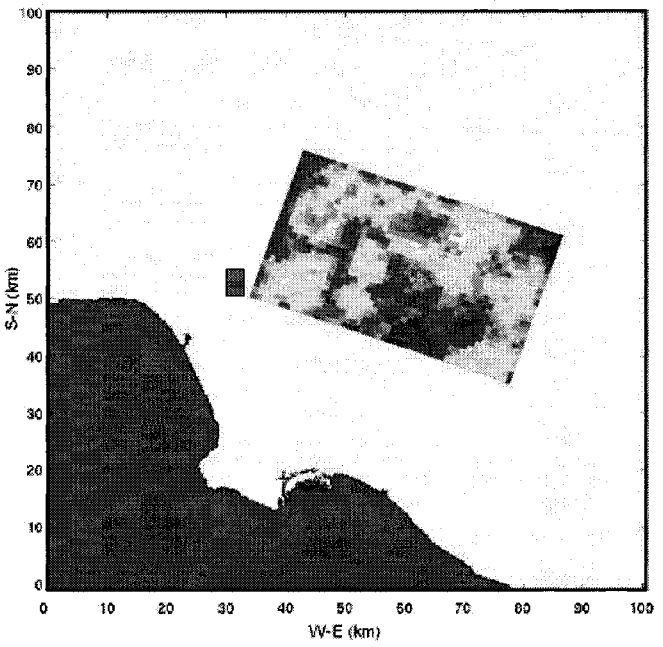

Figure 3.2. In red is the approximate location of the region of interest. Also shown is the Puente Hills fault with the slip distribution for this scenario

Mass proportional Rayleigh damping was used to represent energy dissipation in the soil. It is based on the available geological data for the region by Olsen et al. (2003), with the following assumption for the damping ratio:

$$\zeta = \begin{cases} 25/V_S \text{ (m/sec)} & \text{if } V_S \leq 1500 \text{ m/sec} \\ 5/V_S \text{ (m/sec)} & \text{otherwise} \end{cases} \tag{3.6}$$

The coefficient for the Rayleigh damping α as defined in Chopra (2001) is computed by least squares minimization using Q_S as a target:

$$\alpha = 7.2492\zeta \tag{3.7}$$

and finally the damping matrix is computed as:

$$C = \alpha M \tag{3.8}$$

The scenario for which the simulation was conducted has the following properties:

- Hypocenter global coordinates (X,Y,Z) : (55483.080583, 71728.404324, 10580.420445)
- Type of source : plane
- Moment magnitude : 7.109162
- Extended hypocenter along strike (m) : 11500
- Extended hypocenter down dip (m) : 18900
- Extended strike angle (deg) : 289
- Extended dip angle (deg) : 27
- Extended average rupture velocity (m/sec) : 2800
- Source function type : quadratic
- Average rise time (sec) : 0.5

The DRM box that models the ROI is shown in Fig. 3.3. The input motion at the boundaries, was provided at the exterior surfaces of the box shown in Fig. 3.3, which essentially is the Γ_e bounding surface of Fig. 2.2. It was assumed that along the interface Γ_e - Γ there was no spatial variation of the motion. The input was provided in 6 ASCII files containing velocities, one file for each face for faces 1 to 4 and 2 files for face 5 due to size limitations of the file. There where 70080 nodal locations, each with 3001 time steps with 3 entries per time step which are the velocities in X,Y and Z directions. The file sizes sum to approximately 6.7Gb. The critical importance of the PlaneDRMInputHandler class, and the necessity for very efficient resource management becomes immediately evident due to the need to handle a massive amount of data just to generate the effective seismic input for each time step of the dynamic analysis.

The goal in the initial analysis with the DRM is to match the wavefield within the region of interest (ROI) with the one from the regional analysis that generated the input

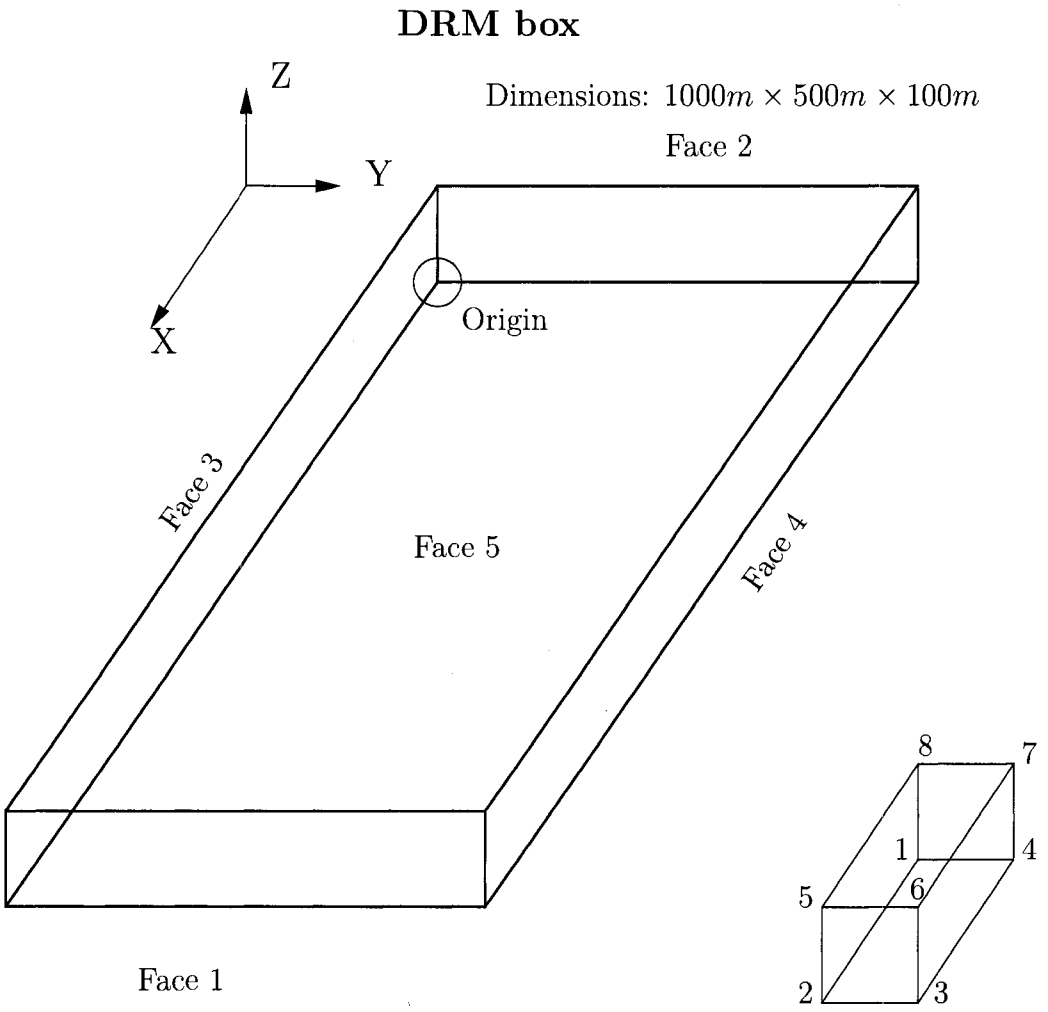

Figure 3.3. DRM box for region of interest, and local element numbering. The origin is 100 m below (along the Z axis) the left right point of the red box shown in Fig. 3.2

for the box. next, a comparison between response due to low frequency pulse type input versus a broadband input is performed. Finally, the same type of comparison is done with a modified soil profile that has roughly 50% of the V_{S30} of the original soil profile, to examine site response effects.

3.2.1 Buffer zone effect

Prior to discussing the simulation results it is necessary to examine the use of buffer zone for this type of simulation. The DRM theory states that if the mesh used for the generation of the input (i.e Step I) is identical to that used for the analysis of the subdomain (i.e Step II) and the materials are linear elastic, then the wave field within the ROI is replicated exactly. This means that the scattered wave $\mathbf{w_e}$ in Eq. (2.6) is zero. In this analysis the meshes differ because of the way they were generated and the assumptions about the soil profile, and also the assumption of no spatial variation within the $\Gamma - \Gamma_e$ layer. Each of these differences, although small, contribute to the fact that the scattered wave field is non-zero. To dissipate the scattered wave field, a buffer zone and absorbing boundary conditions are used, as discussed in Section 2.1.3. To improve the efficacy of the absorbing boundary conditions it is advised to place them at a certain distance away from the domain of interest. While there is no closed form solution for the size of the region that needs to be added as a buffer zone, the issue can be addressed by performing numerical simulations with increasing buffer zone sizes until the model performance is satisfactory.

Three cases were studied with increasing buffer zone sizes. A 0 m a 200 m and a 400 m buffer zone per side per direction were considered. Fig. 3.4 shows a comparison between the cases studied. As the buffer zone size increases the solution converges to the one generated by the analysis providing the DRM input as expected. Based on this comparison, the 400 m buffer zone per side of the ROI was considered satisfactory. The buffer zone improves resolution but at a significant computational cost. Note that for the

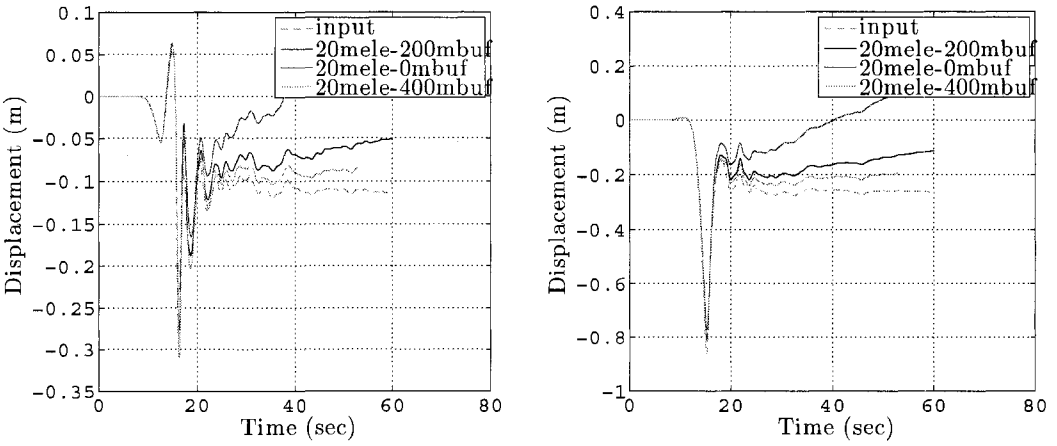

Figure 3.4. Buffer zone size effect on X(left) and Y(right) component of displacement on top surface centernode of ROI

ROI itself with a 20 m element mesh a total of 6250 FLBricks are needed. The same model with the buffer zone goes to 146250 FLBricks a 22 times increase in the model size.

3.2.2 Verification of simulation via comparison at centernode

The comparison between the simulation results at the centernode for the low frequency input simulation against the original the provided the input is described. The results Fig. 3.5 through Fig. 3.8 present a very satisfactory agreement given the complexity of the problem and the differences previously mentioned. These results validate the consistence and stability of the solutions from the developed software. Next, results from the study of the seismic response of the ROI are presented. The differences in the spectral content in the Y component of the centernode can partially be attributed to: as Fig. 3.7 shows there is a 17.5% difference in the input accelerations which would certainly reduce the difference between the two spectral curves and also the differences in damping and mesh are also contributing to the observed difference.

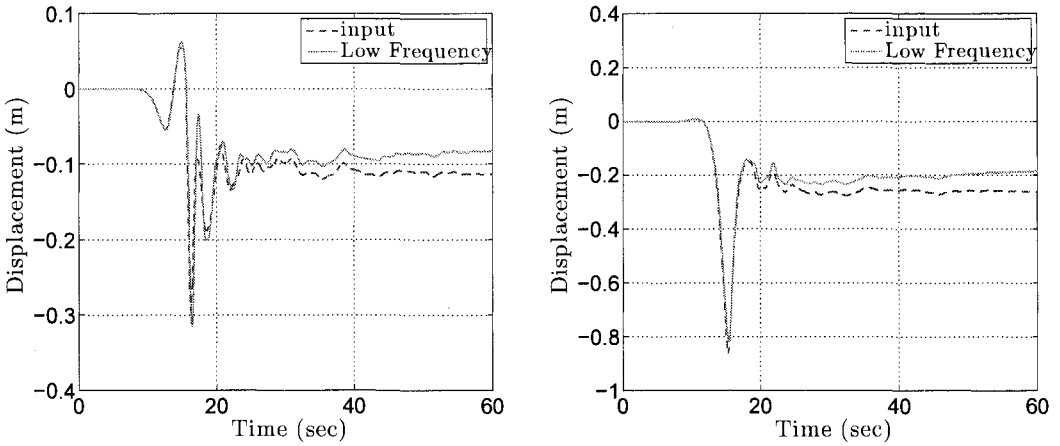

Figure 3.5. Comparison of X(left) and Y(right) component of displacement at top surface centernode

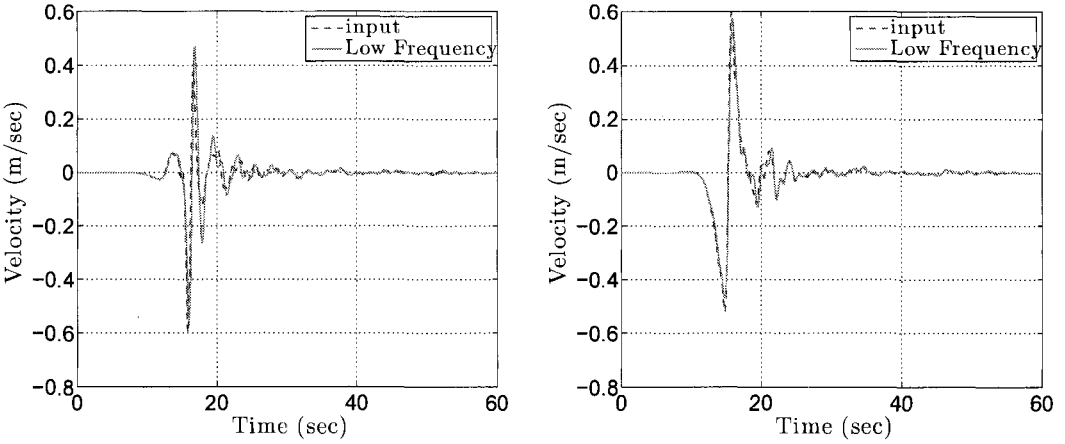

Figure 3.6. Comparison of X(left) and Y(right) component of velocity at top surface centernode

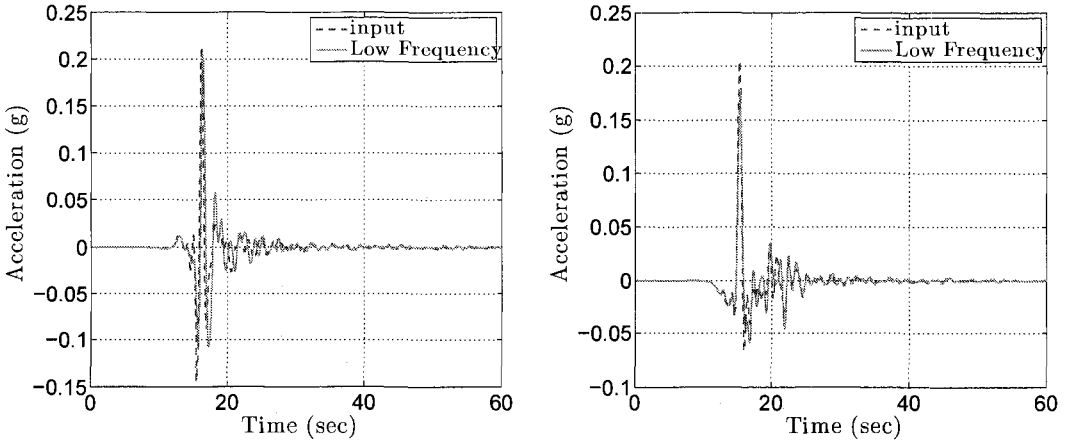

Figure 3.7. Comparison of X(left) and Y(right) component of acceleration at top surface centernode

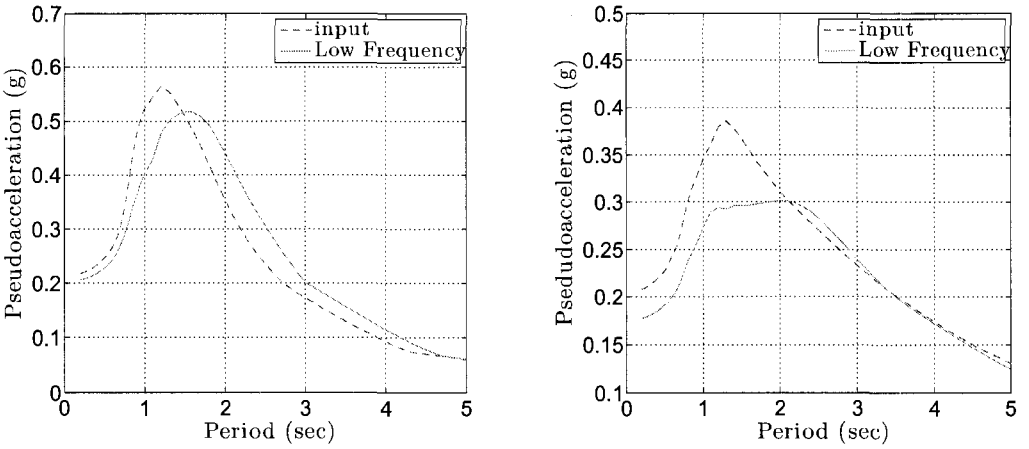

Figure 3.8. Comparison of X(left) and Y(right) component of 5% damped spectrum at top surface centernode

3.2.3 Simulation results for original soil profile using low frequency input

The properties of the soil in the region of interest are given in Table 3.1. The results are given in two centerlines along x and y on the top surface of the ROI in Fig. 3.9 through Fig. 3.12. An observation that assists in comprehending the surface centerline plots is relates to the end points. Where as in every point the quantity plotted is total wave field, at the end points which belong to Γ_e the quantity plotted is the scattered wave field $\mathbf{w_e}$ according to Eq. (2.6). This is a property of the method and not a choice. In order to obtain the total wave field in the end points it is necessary to add the input wave field to it according to Eq. (2.6). An additional means of establishing the correctness of the analysis is that the $\mathbf{w}_e$ wave field is zero or relatively smaller compared to the total wave field. In the results presented is comparatively small, it is not zero because of the differences in the meshes and the assumption of no variation between Γ_e and Γ.

3.2.4 Simulation results for original soil profile using broadband input

The properties of the soil in the region of interest are given in Table 3.1. First a comparison of the wave field at the centernode of the surface of the ROI between the low

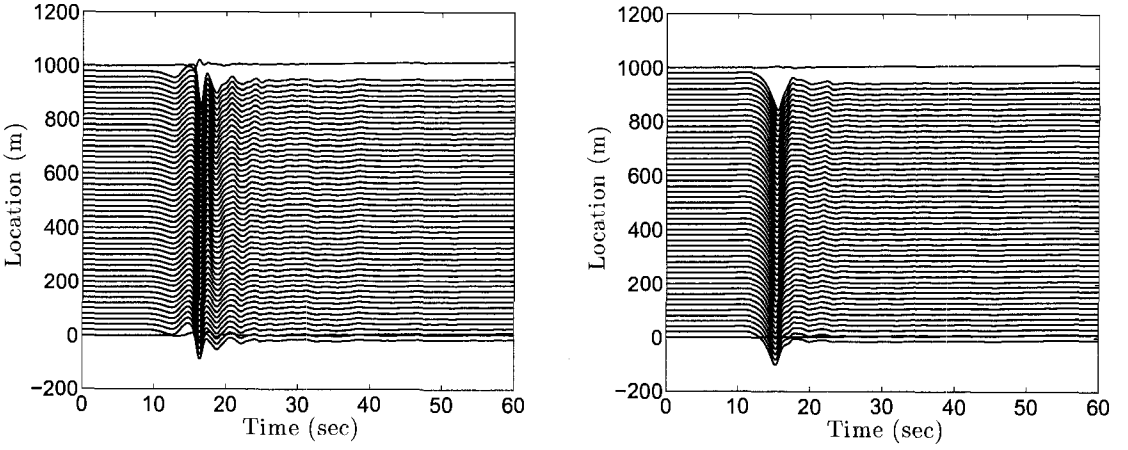

Figure 3.9. X(left) and Y(right) component of ground displacement along X centerline of surface of ROI, for original soil profile with low frequency input

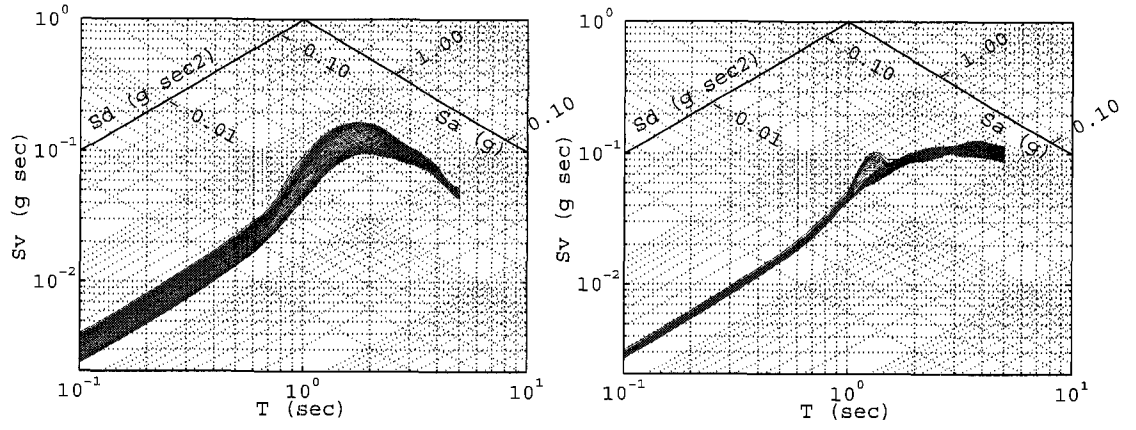

Figure 3.10. X(left) and Y(right) components of 5% damped spectra along X centerline of surface of ROI, for original soil profile with low frequency input

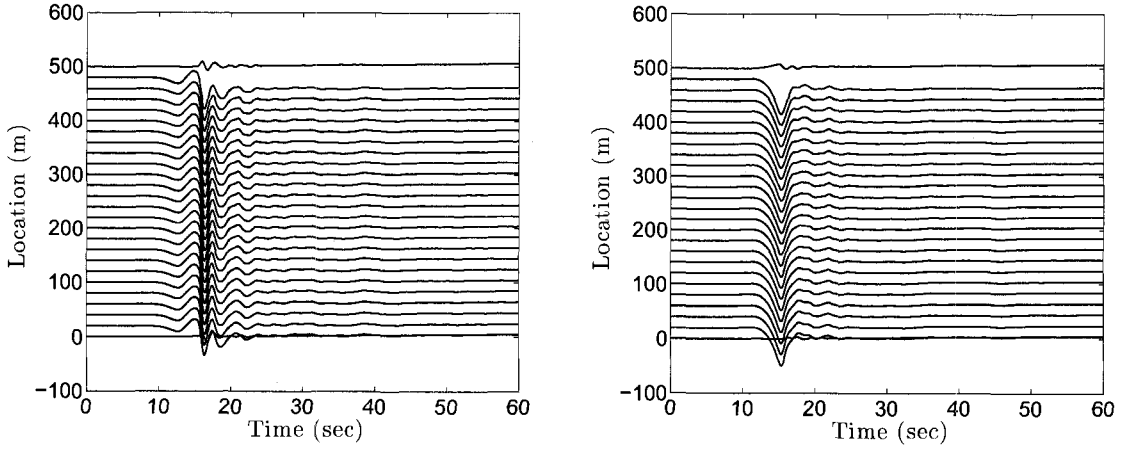

Figure 3.11. X(left) and Y(right) component of ground displacement along Y centerline of surface of ROI, for original soil profile with low frequency input

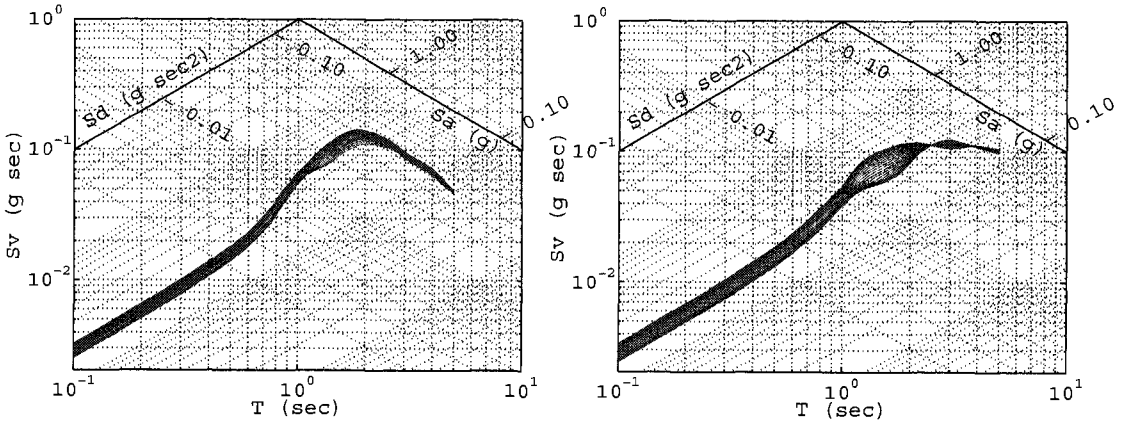

Figure 3.12. X(left) and Y(right) components of 5% damped spectra along Y centerline of surface of ROI, for original soil profile with low frequency input

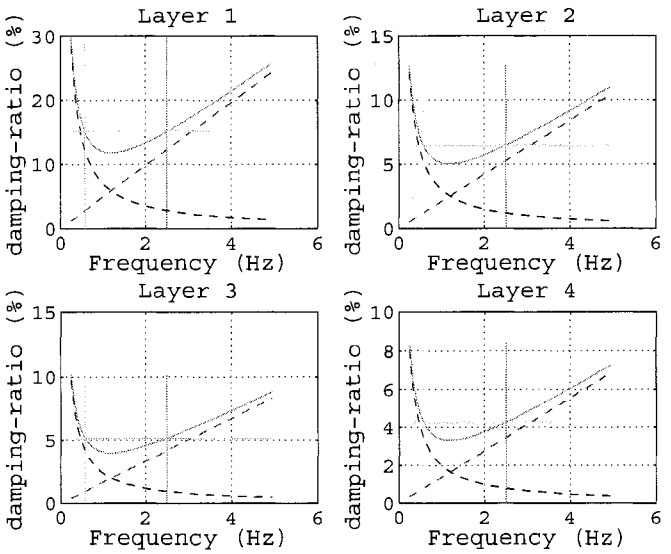

Figure 3.13. Damping ratios for original soil profile

frequency and the broadband is presented. Fig. 3.13 presents the damping ratios used on the 4 layers of the mesh. The results are given in two centerlines along x and y on the top surface of the ROI in Fig. 3.18 through Fig. 3.21. An immediate observation following from Fig. 3.19 and Fig. 3.21 is that the spectra appears to contain to distinct sets of peaks the low and high frequency ones. The two sets of peaks seem to lack continuity though and this is a result of the superposition of the low frequency wave field generated deterministically and the high frequency one generated stochastically. This is a deficiency of the methodology which was inevitable at the time. Eventually the computational capabilities will allow sufficient fidelity simulations at the scale required for the proper wave field generation.

3.2.5 Simulation results for soft soil profile using low frequency input

The soft soil profile has a modified layer profile. The top 30 m are a uniform layer of $V_S = 160$ m/sec. This is effectively a V_S one half of the V_S of the regular soil profile. The remaining 70 m have one half the properties of the corresponding layer of the regular soil

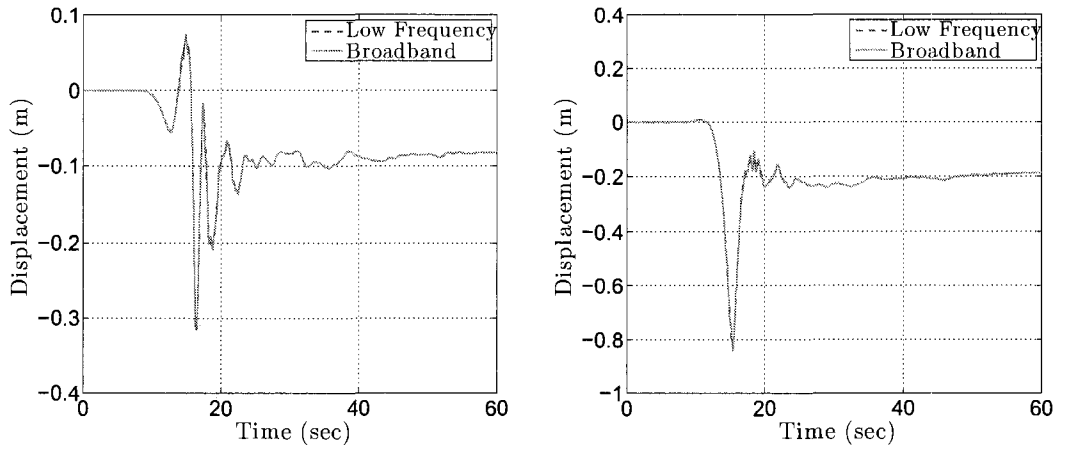

Figure 3.14. Comparison of X(left) and Y(right) component of displacement at top surface centernode

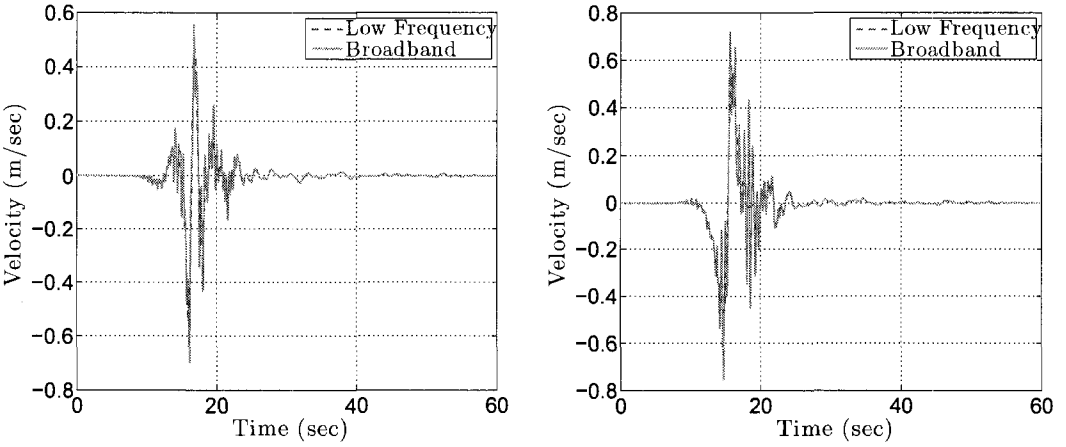

Figure 3.15. Comparison of X(left) and Y(right) component of velocity at top surface centernode

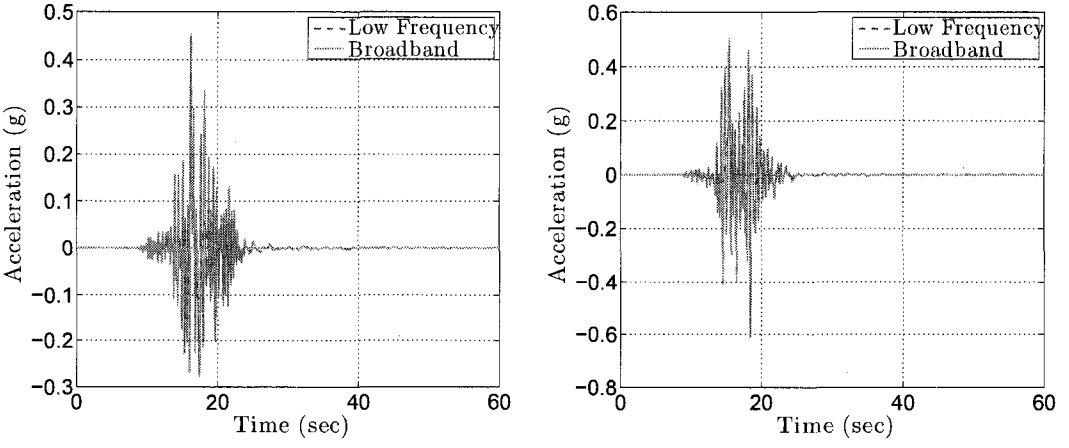

Figure 3.16. Comparison of X(left) and Y(right) component of acceleration at top surface centernode

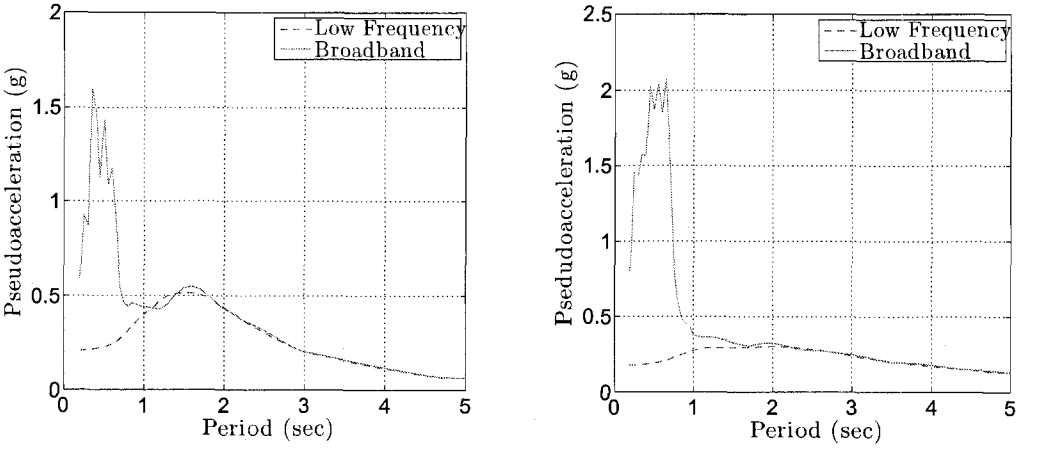

Figure 3.17. Comparison of X(left) and Y(right) component of 5% damped spectrum at top surface centernode

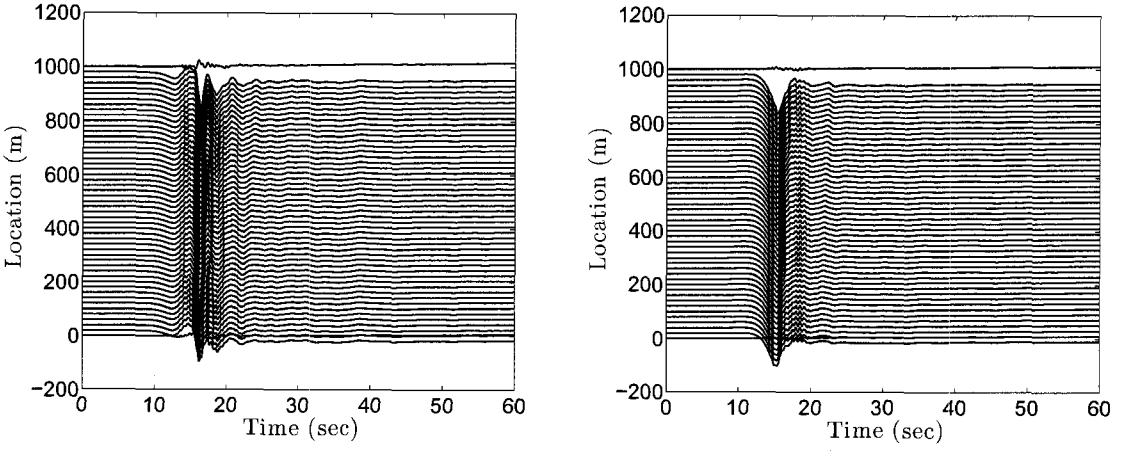

Figure 3.18. X(left) and Y(right) component of ground displacement along X centerline of surface of ROI, for original soil profile with broadband input

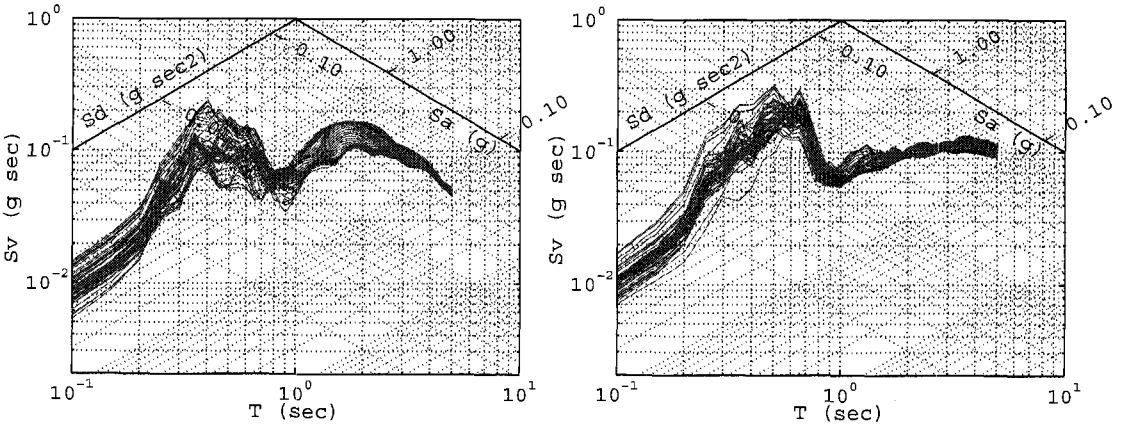

Figure 3.19. X(left) and Y(right) components of 5% damped spectra along X centerline of surface of ROI, for original soil profile with broadband input

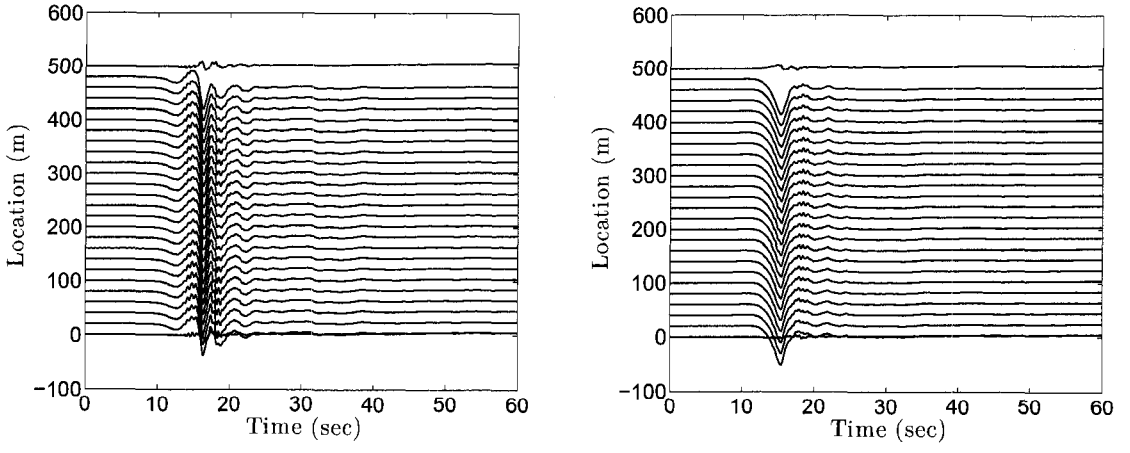

Figure 3.20. X(left) and Y(right) component of ground displacement along Y centerline of surface of ROI, for original soil profile with broadband input

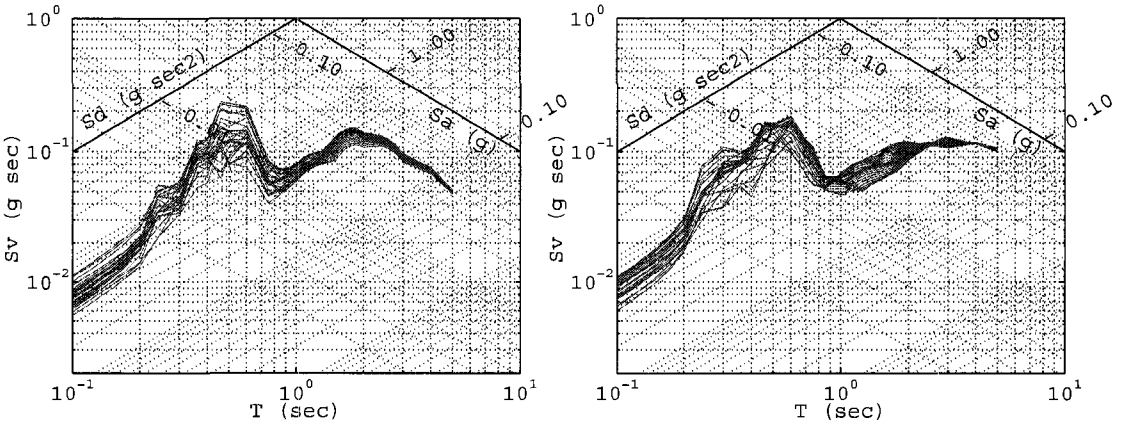

Figure 3.21. X(left) and Y(right) components of 5% damped spectra along Y centerline of surface of ROI, for original soil profile with broadband input

profile. The results on the surface centernode comparing the regular soil response to the soft soil response are shown in Fig. 3.22 through Fig. 3.25. The surface centerline plots are shown in Fig. 3.26 through Fig. 3.29. By comparing the latter plots to the regular soil profile and especially the spectral plots it can be seen that an increase in the spatial variability is shown. The breadth of the spectral plots is wider for the soft soil case.

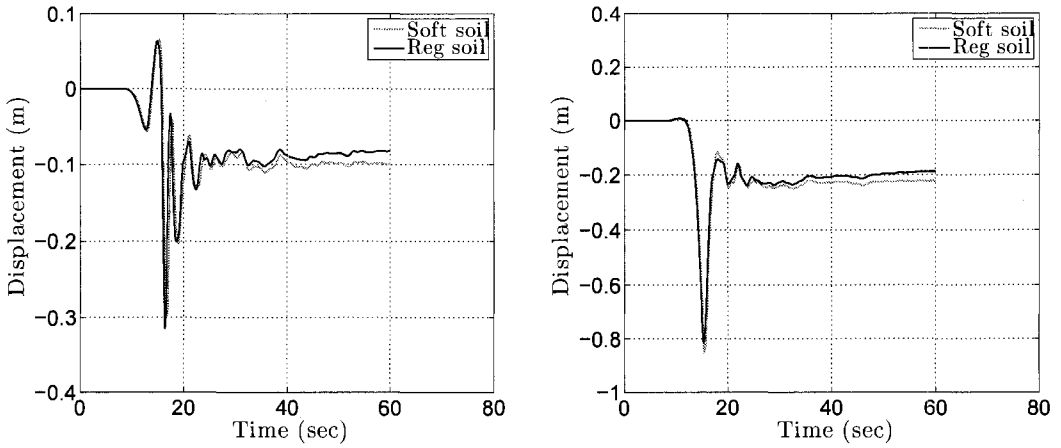

Figure 3.22. Comparison of X(left) and Y(right) component of displacement at top surface centernode

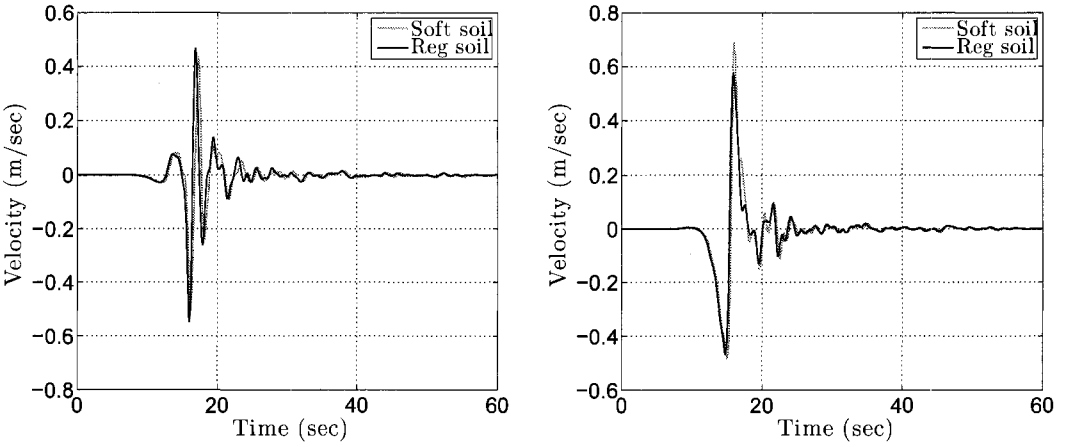

Figure 3.23. Comparison of X(left) and Y(right) component of velocity at top surface centernode

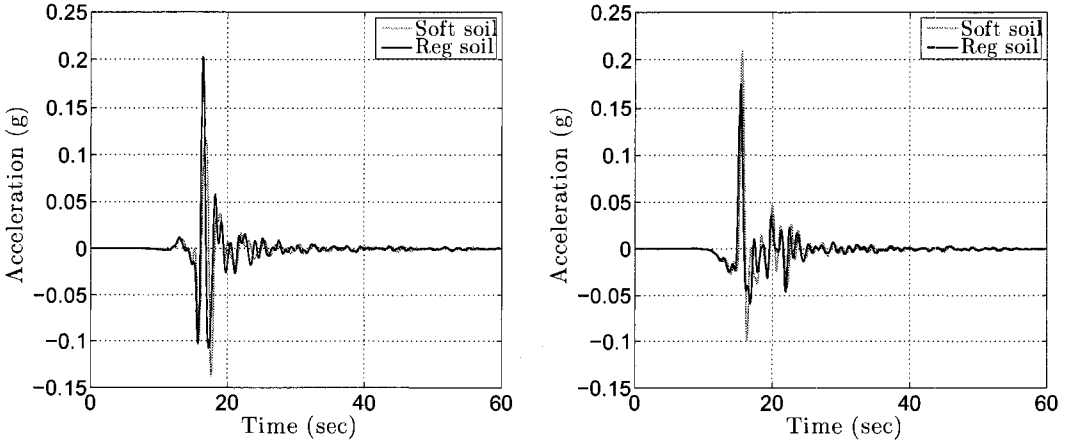

Figure 3.24. Comparison of X(left) and Y(right) component of acceleration at top surface centernode

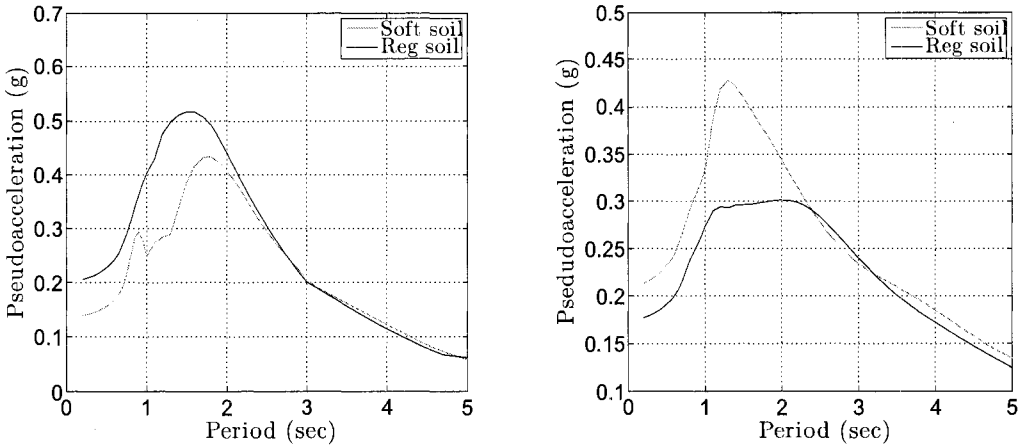

Figure 3.25. Comparison of X(left) and Y(right) component of 5% damped spectrum at top surface centernode

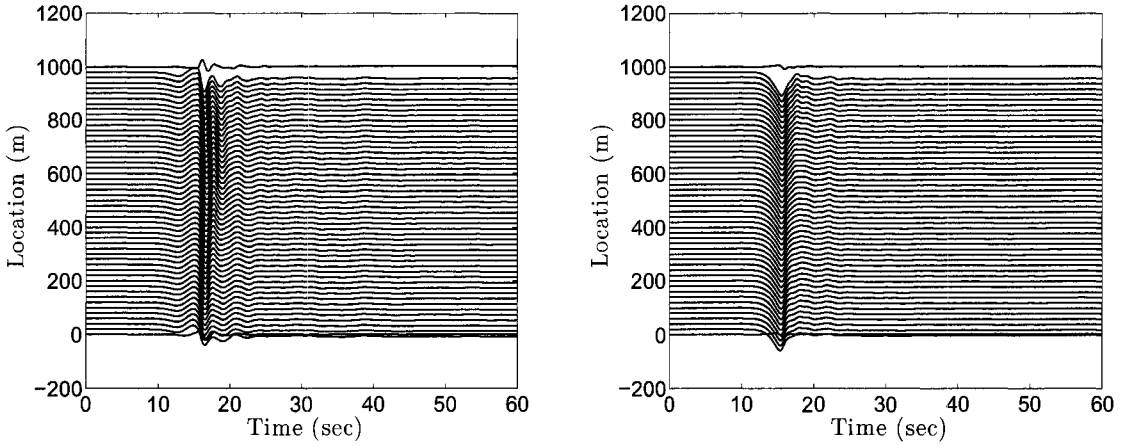

Figure 3.26. X(left) and Y(right) component of ground displacement along X centerline of surface of ROI, for soft soil profile with low frequency input

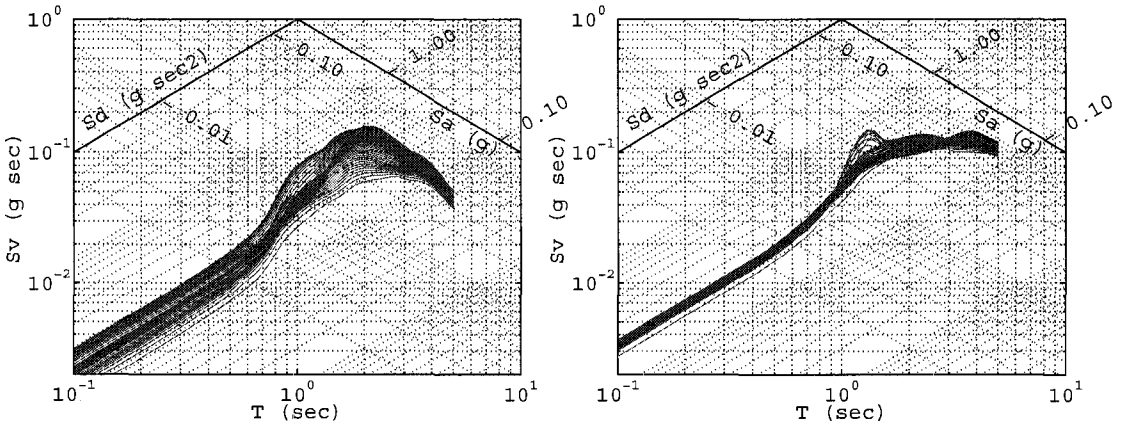

Figure 3.27. X(left) and Y(right) components of 5% damped spectra along X centerline of surface of ROI, for soft soil profile with low frequency input

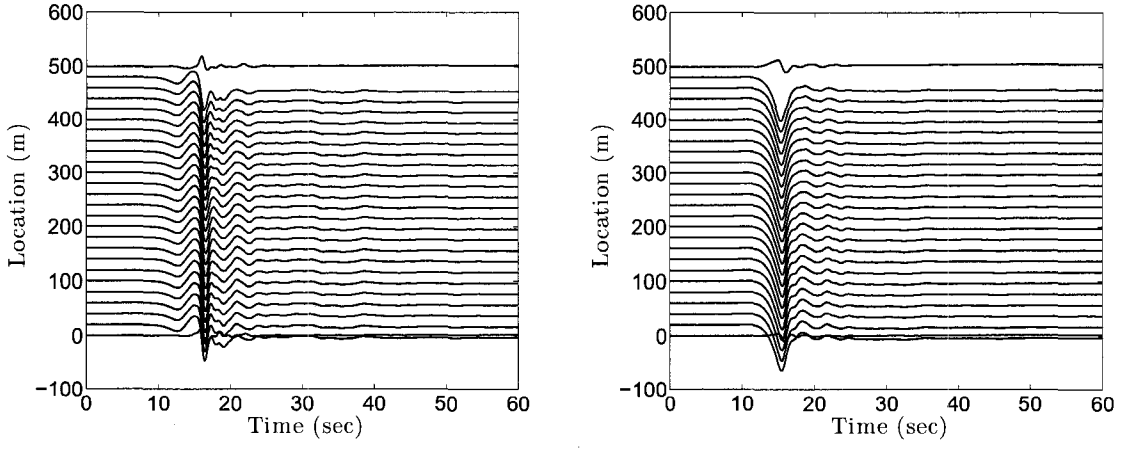

Figure 3.28. X(left) and Y(right) component of ground displacement along Y centerline of surface of ROI, for softsoil soil profile with low frequency input

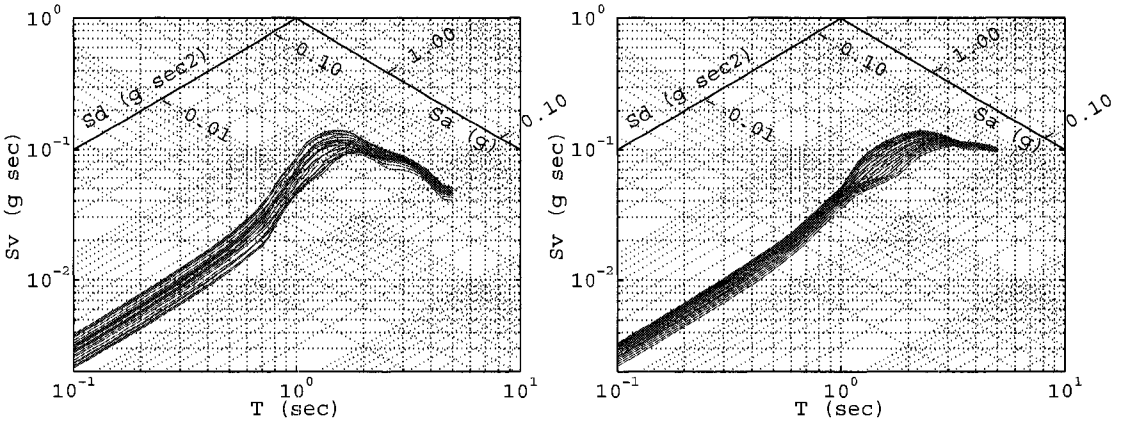

Figure 3.29. X(left) and Y(right) components of 5% damped spectra along Y centerline of surface of ROI, for soft soil profile with low frequency input

Chapter 4

Mixed implicit-explicit time integration of domains

This chapter presents extensions of a mixed explicit-implicit time integration algorithm that allows for non-linear systems in the implicit partition. An efficient software design is presented for introducing mixed integration in the explicit code previously presented. The chapter concludes with verification examples of the parallel implicit-explicit time integration method.

4.1 Background and previous work

The focus of this research so far has been the analysis of linear semi-infinite soil subdomains for which explicit time integration suffices. The analysis of structural models and their foundations coupled with three dimensional soil requires implicit time integration in the solution procedure. The necessity of implicit time integration is dictated from the fact that explicit solution for structures requires very small time steps given the frequencies involved mainly in the axial direction of the beam elements due to their very high stiffness. These frequencies are much higher compared to those of the soil elements and render the

Courant condition for the beam elements hard to satisfy. In addition, beam elements contain rotational degrees of freedom, and assigning masses to those degrees of freedom is a non trivial task. The second reason could be bypassed if one introduces static condensation in the structural model and solves only for the translational degrees of freedom during the analysis. This procedure is cumbersome and would only work for linear structural models. All of the above reasons support the use of implicit time integration.

Introducing implicit integration for the entire coupled model, results in a very large system of equations. Its solution via factorization of the left hand side operator **A** implies large memory requirements, increased floating point operations and significant interprocessor communication if a direct solver is to be used and the size of the problem, presently in the order of hundreds of millions of DOFs, renders this approach infeasible. Iterative solvers on the other hand scale well but still contain more floating point operations than an explicit scheme would require. It is exactly these very high computational requirements and cost of finite element methods that is one of the primary arguments against their use for SFSI problems that has led in the development and use of more simplified methods. For detailed discussion of methods for direct and iterative linear solvers see Demmel (1997); Golub and Van Loan (1996).

In this problem the largest part of the domain is wave propagation through the soil for which explicit time integration works very well. Consequently and in order to maintain reasonable solution times, scalability, efficiency and high performance for the custom application, mixed implicit-explicit time integration is selected for the coupled structure-soil system respectively.

The two classes of time integration algorithms for transient analysis are implicit and explicit. Methods have been developed to achieve a simultaneous combination of the attributes of both classes. Among those, Hughes and Liu (1978a,b) developed an implicit-explicit partition scheme where both partitions are integrated using the same time-step. An approach similar to the previous one was presented by Liu and Belytschko (1982)

and introduced a general mixed time implicit-explicit partition procedure that allows for different time steps and different integration methods to be used in different parts of the finite element mesh. An $mE-I$ partition where m explicit steps are used for each implicit step was presented. The mesh is divided into two groups. The explicit group is integrated with a time-step Δt and the implicit group with a time-step $m\Delta t$. The implicit partition is integrated using the Newmark algorithm (Newmark, 1959), and the explicit partition is integrated using an explicit method based on the Newmark algorithm. An introduction to this explicit method was given in Section 2.2.3. Another work on mixed time integration, using Lagrange multipliers to represent the forces along the interface of the explicit and implicit partition is by Gravouil and Combescure (2001).

4.2 $mE - I$ time integration scheme for linear systems

The algorithmic aspects of the mixed time integration method as given in Liu and Belytschko (1982) are presented here. The elements are partitioned in the E and I groups and consequently the matrices obtained by assembly on each group are denoted using the same superscripts. Thus any global matrix is the sum of the explicit and implicit matrices. $\mathbf{M} = \mathbf{M}^E + \mathbf{M}^I$, $\mathbf{C} = \mathbf{C}^E + \mathbf{C}^I$, $\mathbf{K} = \mathbf{K}^E + \mathbf{K}^I$. Correspondingly, the degrees of freedom are partitioned in E,I and B sets where the latter denotes the interface degrees of freedom; also $B \subset I$. Finally all vectors are assembled as the summation of two vectors, one from the explicit and one from the implicit set of dofs. The solution algorithm is given in Fig. 4.1 and Fig. 4.2.

4.3 $mE - I$ time integration scheme for nonlinear systems

This section generalizes standard mixed time integration procedure defined in the previous section to handle nonlinear material laws for the elements of the implicit partition. For the explicit partition it is obvious that there can always be found a time step

1. Initialization. Set $n = 0$, $\mathbf{u}_0$, $\dot{\mathbf{u}}_0$.

2. Determine $a_o = \mathbf{M}^{-1}(\mathbf{p}_0 - \mathbf{C}\dot{\mathbf{u}}_0 - \mathbf{K}\mathbf{u}_0)$

3. Form and factorize $\mathbf{H}$.

$$\begin{cases} \mathbf{H} = \mathbf{H}^E + \mathbf{H}^I \\ \mathbf{H}^E = \mathbf{M}^E \\ \mathbf{H}^I = \mathbf{M}^I + \gamma m \Delta t \mathbf{C}^I + \beta m^2 \Delta t^2 \mathbf{K}^I \end{cases} \tag{4.1}$$

4. **for j=1,...,m** repeat:

5. Define

$$\begin{cases} \tilde{\mathbf{u}}^E_{n+j} = \mathbf{u}^E_{n+j-1} + \Delta t \dot{\mathbf{u}}^E_{n+j-1} + (\frac{1}{2} - \beta)\Delta t^2 \ddot{\mathbf{u}}^E_{n+j-1} \\ \tilde{\dot{\mathbf{u}}}^E_{n+j} = \dot{\mathbf{u}}^E_{n+j-1} + (1-\gamma)\Delta t \ddot{\mathbf{u}}_{n+j-1} \end{cases} \tag{4.2}$$

6. Define

$$\begin{cases} \tilde{\mathbf{u}}^I_{n+j} = \mathbf{u}^I_n + j\Delta t \dot{\mathbf{u}}^I_n + (\frac{1}{2} - \beta) j^2 \Delta t^2 \ddot{\mathbf{u}}^I_n \\ \tilde{\dot{\mathbf{u}}}^I_{n+j} = \dot{\mathbf{u}}^I_n + (1-\gamma) j \Delta t \ddot{\mathbf{u}}^I_n \end{cases} \tag{4.3}$$

On the subset $"B"$ for $1 \leq j < m$ and on $"I"$ for $j = m$.

7. Form:

$$\begin{cases} \mathbf{g}^E_{n+j} = \mathbf{M}^E \tilde{\mathbf{u}}_{n+j} + \beta \Delta t^2 \mathbf{W}(\mathbf{p}^E_{n+j} - \mathbf{C}^E \tilde{\dot{\mathbf{u}}}_{n+j} - \mathbf{K}^E \tilde{\mathbf{u}}_{n+j}) \\ \mathbf{g}^I_{n+m} = \beta m^2 \Delta t^2 \mathbf{p}^I_{n+m} + \mathbf{M}^I \tilde{\mathbf{u}}_{n+m} - \beta m^2 \Delta t^2 \mathbf{C}^I \tilde{\dot{\mathbf{u}}}_{n+m} + \gamma m \Delta t \mathbf{C}^I \tilde{\mathbf{u}}_{n+m} \end{cases} \tag{4.4}$$

Where: $\tilde{\mathbf{u}}_{n+m} = \tilde{\mathbf{u}}^E_{n+m} + \tilde{\mathbf{u}}^I_{n+m}$ and $\tilde{\dot{\mathbf{u}}}_{n+m} = \tilde{\dot{\mathbf{u}}}^E_{n+m} + \tilde{\dot{\mathbf{u}}}^I_{n+m}$ and also

$$\mathbf{W} = \begin{bmatrix} m^2\mathbf{I} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \tag{4.5}$$

The first $|I|$ columns and rows of $\mathbf{W}$ are diagonal and equal to m^2 and the subsequent $|E|$ is an identity matrix. Here E, I refer to the nodal partitions.

Figure 4.1. $mE - I$ pseudocode for linear systems by Liu and Belytschko (1982), part A

8. Solve for:

$$\begin{cases} \mathbf{H}\mathbf{u}_{n+m} = \mathbf{g}_{n+m} & \text{if } j = m \\ \mathbf{H}^E \mathbf{u}^E_{n+m} = \mathbf{g}^E_{n+j} & \text{if } 1 \leq j < m \end{cases} \tag{4.6}$$

Where $\mathbf{u}_{n+m} = \mathbf{u}^E_{n+m} + \mathbf{u}^I_{n+m}$ and $\mathbf{g}_{n+m} = \mathbf{g}^E_{n+m} + \mathbf{g}^I_{n+m}$.

9. Determine:

$$\begin{cases} \ddot{\mathbf{u}}^E_{n+j} = (\mathbf{u}^E_{n+j} - \tilde{\mathbf{u}}^E_{n+j})/(\beta \Delta t^2) \\ \dot{\mathbf{u}}^E_{n+j} = \tilde{\dot{\mathbf{u}}}^E_{n+j} + \gamma \Delta t \ddot{\mathbf{u}}^E_{n+j} \end{cases} \tag{4.7}$$

10. If $j = m$ determine:

$$\begin{cases} \ddot{\mathbf{u}}^I_{n+m} = (\mathbf{u}^I_{n+m} - \tilde{\mathbf{u}}^I_{n+m})/(\beta m^2 \Delta t^2) \\ \dot{\mathbf{u}}^I_{n+m} = \tilde{\dot{\mathbf{u}}}^I_{n+m} + \gamma m \Delta t \ddot{\mathbf{u}}^I_{n+m} \end{cases} \tag{4.8}$$

and form $\ddot{\mathbf{u}}_{n+m}, \dot{\mathbf{u}}_{n+m}$.

11. **If $j < m$ set $j = j + 1$ and go to 4.**

12. **Check for termination of analysis or increment $n = n + m$ and go to 4.**

Figure 4.2. $mE - I$ pseudocode for linear systems by Liu and Belytschko (1982), Part B

Δt for which the method will converge in one step. In the implicit partition nonlinear elements are present and thus a root-finding method such as the Newton-Raphson will be used. Since the iterations these algorithms perform are incremental it is worth reformulating the equations of the previous section. For computational efficiency the elements are repartitioned and a third subset $E_B \subset E$ is created. The elements that belong to the subset E_B are explicit elements that contain dofs that belong to the subset B which is the set of interface degrees of freedom. These elements even though they are integrated explicitly will need to perform more than one pass per time step to compute the element forces for the iterative solution of the total equation of motion on the interface degrees of freedom. The solution algorithm is presented in Fig. 4.3 and Fig. 4.4.

1. Initialization. Set $n = 0$, $\mathbf{u}_0$, $\dot{\mathbf{u}}_0$.

2. Determine $\mathbf{a}_0 = \mathbf{M}^{-1}(\mathbf{p}_0 - \mathbf{C}\dot{\mathbf{u}}_0 - \mathbf{K}\mathbf{u}_0)$

3. Form:

$$\mathbf{H}^E = \mathbf{M}^E \tag{4.9}$$

4. **for j=1,...,m** repeat:

5. Define

$$\begin{cases} \tilde{\mathbf{u}}^E_{n+j} = \mathbf{u}^E_{n+j-1} + \Delta t \dot{\mathbf{u}}^E_{n+j-1} + (\frac{1}{2} - \beta)\Delta t^2 \ddot{\mathbf{u}}^E_{n+j-1} \\ \tilde{\dot{\mathbf{u}}}^E_{n+j} = \dot{\mathbf{u}}^E_{n+j-1} + (1-\gamma)\Delta t \ddot{\mathbf{u}}^E_{n+j-1} \\ \tilde{\ddot{\mathbf{u}}}^E_{n+j} = -\frac{1}{\beta \Delta t}\dot{\mathbf{u}}^E_{n+j-1} + (1 - \frac{1}{2\beta})\ddot{\mathbf{u}}^E_{n+j-1} \end{cases} \tag{4.10}$$

6. Define

$$\begin{cases} \tilde{\dot{\mathbf{u}}}^I_{n+j} = \dot{\mathbf{u}}^I_n + (1-\gamma) j \Delta t \ddot{\mathbf{u}}^I_n \\ \tilde{\ddot{\mathbf{u}}}^I_{n+j} = -\frac{1}{\beta j \Delta t}\dot{\mathbf{u}}^I_n + (1 - \frac{1}{2\beta})\ddot{\mathbf{u}}^I_n \end{cases} \tag{4.11}$$

On the subset $"B"$ for $1 \leq j < m$ and on $"I"$ for $j = m$.

Also on the subset B define:

$$\tilde{\mathbf{u}}^I_{n+j} = \mathbf{u}^I_n + j\Delta t \dot{\mathbf{u}}^I_n + (\frac{1}{2} - \beta) j^2 \Delta t^2 \ddot{\mathbf{u}}^I_n \tag{4.12}$$

The above equation is to be used by the elements belonging in the E_B subset. For the solution on the I elements there is no displacement predictor, as is the case for the classical Newmark algorithm.

7. Form:

$$\begin{cases} \mathbf{g}^E_{n+j} = \beta \Delta t^2 \mathbf{W}(\mathbf{p}^E_{n+j} - \mathbf{K}^E \tilde{\mathbf{u}}_{n+j} - \mathbf{C}^E \tilde{\dot{\mathbf{u}}}_{n+j} - \mathbf{M}^E \tilde{\ddot{\mathbf{u}}}_{n+j}) \\ \mathbf{g}^I_{n+m} = \beta m^2 \Delta t^2 (\mathbf{p}^I_{n+m} - \mathbf{K}^I \tilde{\mathbf{u}}_{n+m} - \mathbf{C}^I \tilde{\dot{\mathbf{u}}}_{n+m} - \mathbf{M}^I \tilde{\ddot{\mathbf{u}}}_{n+m}) \end{cases} \tag{4.13}$$

Where: $\tilde{\mathbf{u}}_{n+m} = \tilde{\mathbf{u}}^E_{n+m} + \tilde{\mathbf{u}}^I_{n+m}$ and $\tilde{\dot{\mathbf{u}}}_{\mathbf{n+m}} = \tilde{\dot{\mathbf{u}}}^{\mathbf{E}}_{\mathbf{n+m}} + \tilde{\dot{\mathbf{u}}}^{\mathbf{I}}_{\mathbf{n+m}}$.

The matrix $\mathbf{W}$ is defined as in the previous case.

Figure 4.3. Incremental $mE - I$ formulation pseudocode for nonlinear systems, Part A

8. Solve for:

$$\mathbf{H}^E \Delta \mathbf{u}^E_{n+j} = \mathbf{g}^E_{n+j} \tag{4.14}$$

Update :

$$\begin{cases} \ddot{\mathbf{u}}^E_{n+j} = \tilde{\ddot{\mathbf{u}}}^E_{n+j} + \frac{1}{\beta \Delta t^2} \Delta \mathbf{u}^E_{n+j} \\ \mathbf{u}^E_{n+j} = \mathbf{u}^E_{n+j-1} + \Delta \mathbf{u}^E_{n+j} \\ \dot{\mathbf{u}}^E_{n+j} = \tilde{\dot{\mathbf{u}}}^E_{n+j} + \gamma \Delta t \ddot{\mathbf{u}}^E_{n+j} \end{cases} \tag{4.15}$$

If $j = m$ and **while** (*convergence criterion unsatisfied*)

$$\begin{aligned} \mathbf{H}^I &= \mathbf{M}^I + (\gamma m \Delta t)\mathbf{C}^I + (\beta m^2 \Delta t^2)\mathbf{K}^I_T \\ \mathbf{g}^I_{n+m} &= \beta m^2 \Delta t^2 (\mathbf{p}^I_{n+m} - \mathbf{K}^I \tilde{\mathbf{u}}_{n+m} - \mathbf{C}^I \tilde{\dot{\mathbf{u}}}_{n+m} - \mathbf{M}^I \tilde{\ddot{\mathbf{u}}}_{n+m}) \\ \mathbf{g}^{E_B}_{n+m} &= \beta \Delta t^2 \mathbf{W} (\mathbf{p}^{E_B}_{n+m} - \mathbf{K}^{E_B} \tilde{\mathbf{u}}_{n+m} - \mathbf{C}^{E_B} \tilde{\dot{\mathbf{u}}}_{n+m} - \mathbf{M}^{E_B} \tilde{\ddot{\mathbf{u}}}_{n+m}) \\ (\mathbf{H}^I + \mathbf{H}^{E_B}) \Delta \mathbf{u}^I_{n+m} &= \mathbf{g}_{n+m} \\ \mathbf{u}^I_{n+m} &= \mathbf{u}^I_n + \Delta \mathbf{u}^I_{n+m} \\ \ddot{\mathbf{u}}^I_{n+m} &= (\mathbf{u}^I_{n+m} - \tilde{\mathbf{u}}^I_{n+m})/(\beta m^2 \Delta t^2) \\ \dot{\mathbf{u}}^I_{n+m} &= \tilde{\dot{\mathbf{u}}}^I_{n+m} + \gamma m \Delta t \ddot{\mathbf{u}}^I_{n+m} \end{aligned} \tag{4.16}$$

Where: $\tilde{\mathbf{u}}_{n+m} = \tilde{\mathbf{u}}^{E_B}_{n+m} + \tilde{\mathbf{u}}^I_{n+m}$, $\tilde{\dot{\mathbf{u}}}_{n+m} = \tilde{\dot{\mathbf{u}}}^{E_B}_{n+m} + \tilde{\dot{\mathbf{u}}}^I_{n+m}$ and $\tilde{\ddot{\mathbf{u}}}_{n+m} = \tilde{\ddot{\mathbf{u}}}^{E_B}_{n+m} + \tilde{\ddot{\mathbf{u}}}^I_{n+m}$. The implicit predictors are to be recomputed at each iteration using the equations of step 6. The unbalanced forces do not need to be computed for the first iteration due to step 7. The superscript E_B emphasizes that the explicit unbalanced computations need to be performed only on the E_B elements.

9. **If** $j < m$ **set** $j = j + 1$ **and go to 4.**

10. **Check for termination of analysis or increment** $n = n + m$ **and go to 4.**

Figure 4.4. Incremental $mE - I$ formulation pseudocode for nonlinear systems, Part B

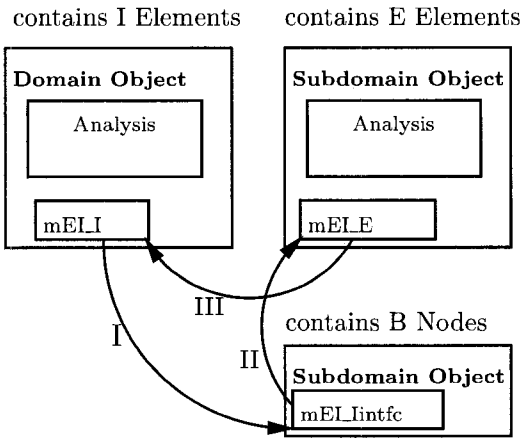

Figure 4.5. Schematic design for serial processing of the implementation of the $mE - I$ scheme (arrow indicates communication of information)

4.4 Stability of the $mE - I$ method

The stability of the scheme is presented in a numerical fashion in Liu and Belytschko (1982). Hughes and Liu (1978a) present a stability analysis of the $E - I$ method i.e. for m=1. Their proof uses the energy method. For a discussion of the energy method the reader is referred to Richtmyer and Morton (1957). Hughes et al. (1979) discuss stability related to mixed integration methods. Hughes (2000) also discusses the stability derivations of the Newmark algorithm, predictor-corrector algorithms and the explicit-implicit scheme for $m = 1$.

4.5 Implementation of the $mE - I$ scheme for nonlinear systems

This section describes the design and implementation of the $mE - I$ scheme. The design fully utilizes the explicit code developed in Chapter 2. The essential design concept is that the explicit part of the mesh is partitioned among several processors and then in each processor analyze several structures that are to be integrated implicitly, and which

are to communicate with the explicit partition of the same process via a unified interface. This communication between the explicit and implicit partition is within the same address space or over distributed address spaces. For the purposes of this work both objects will be residing in the same address space. The design allows for subclassing and thus various implementations of the virtual method that handles the communication scheme and hence allow for various communication schemes. Finally, an important goal is for all the communication needed for the parallel computation for the wave propagation in the soil to be with respect to the explicit part of the mesh only. This will preserve the balance in the communication graph which is essential for scalability, and will allow for utilization of the components designed and implemented in Chapter 2. This means that no implicit explicit interface dof will coincide with interface nodes with respect to the partition of the soil. This approach will lead to maintaining the scalability we demonstrated previously even though there will be increased load imbalance since not all processors are to have implicit partitions as well.

Fig. 4.5 shows the objects resident in memory for the case of one processor where we analyze a domain using mixed $mE - I$ integration, one Domain and two Subdomain objects. The choice of having the implicit elements I in the Domain object, and the explicit elements E in a Subdomain object (I,E are with respect to the solution algorithm presented previously) is dictated by the order in which a Domain object that is undergoing analysis is integrated. In a Domain object, the FE elements of the object are integrated after any Subdomains objects have completed their integration. The m^{th} step of the loop in the implicit elements, requires the m^{th} step solution of the explicit elements, to assemble the unbalanced forces on the interface degrees of freedom via accounting for the contribution of the E_B elements according to Eq. (4.16). Consequently the ordering selected is unique and follows the algorithmic order.

The creation of an additional Subdomain object that contains the interface degrees of freedom is dictated by the fact that the E elements require the computation of predictor

values on the interface degrees of freedom B, used to assemble the unbalanced forces on the E elements. These predictors at step $n+j$ at the interface B, require the converged step n at the I elements. It may seem as unnecessary to duplicate the interface nodes B across the two Subdomains, but the duplication allows for copying the n^{th} step values from the implicit interface once at the beginning of the loop of the j steps to the interface Subdomain object, and from there on computation of the predictors for each of the j steps. The explicit elements contain pointers to the nodes of the interface Subdomain, which allow for the breaking of the cyclic dependancies that arise. Otherwise the implicit integrator would have to be declared containing the explicit and vice versa, a technique termed forward declaration in C++, a not good programming practice.

Each of the three arrows in Fig. 4.5 indicate transfer of information between the two objects. The methods that correspond to the I and III arrows are declared virtual in the design so that subclassing is permitted in order for various communication schemes to be implementable. For the scope of this work, note that all three calls are in the same address space. In a more general parallel computation setting the implicit integrator would be residing to a different processor or even be handled by a different MPI communicator, in which case the corresponding methods would be wrapping MPI calls or some other form of communication mechanism.

Given the sequence of execution in the computations, there exists an ordering for the completion of one step in the $mE-I$ algorithm. The implicit integrator at each of the Newton Raphson iterations it performs, pulls essentially from the E_B elements an updated $g_{n+m}^{E_B}$. Also the mEI_I integrator pulls the $\mathbf{H}^{E_B}$ entries that correspond to the B degrees of freedom. This operation is not possible to be performed using the interface offered from OpenSees. While there exists a method to access the right hand side (RHS) of the system of equations there are no access methods for the LHS. The modification of the customSOE class was necessary in order to be able to "extract" components of one system of equations $\mathbf{Ax}=\mathbf{b}$, both from the matrix $\mathbf{A}$ and the vector $\mathbf{b}$ and add

them to the implicit system object. The addition in the implicit system object is done inside the in the **int mEI_I::formTangent()**, **int mEI_I::formUnbalance()** methods using the **virtual Vector& mEI_E::getA()** and **virtual Vector& mEI_E::getb()** calls. The latter uses the **Vector& SOE::getB()** call, but to implement the former, the method **Vector& customSOE::getPartofA(double* A, ID& interface)** was developed. Similar methods can easily be derived for all the SOE classes in OpenSees if needed. They are custom for each class given the different storage schemes used. Note that these two methods: **virtual Vector& mEI_E::getA()** and **virtual Vector& mEI_E::getb()** can be subclassed depending on the communication requirement between the mEI_E and mEI_I classes. In the case of Fig. 4.5 the implementation is for a common address space. Another detail that was handled carefully with performance in mind is the addition of the components of H^{E_B} to the implicit system. Upon completion of the **virtual Vector& mEI_E::getA()** call there is a vector containing the values needed to be added along the diagonal of the implicit **A**. OpenSees only allows for entire matrices to be added in the **void SOE::addA(Matrix& , ID&)** and in order to efficiently add elements along the diagonal only a matrix of size one is created and each element of the received vector is added individually for a total cost of $O(d)$ where d is the number of entries needed to be added. That is as opposed to creating an entire matrix that would have non zeros only on the diagonal and add all of its elements for a total cost of $O(d^2)$. Upon completion and convergence, the mEI_I pushes the newly computed values to the mEI_Iintfc and then the mEI_E proceeds with computing m steps prior to the next mEI_I step.

Finally Fig. 4.5 shows how does the previously present design generalize for parallel processing. More specifically it shows a four processor MPI communicator which is partitioned into two sub communicators, one for the implicit partition of the domain and one for the explicit. Three types of lines can be seen: Dashed lines indicate interprocess communication but also inter sub-communicator communication. Dashed-dotted lines indicate interprocess intra sub-communicator and finally solid lines are within the same address

space. For example the **Vector& mEI_E::getA()** and **Vector& mEI_E::getb()** should be re implemented to account for the use of MPI.

4.6 Verification of the implementation

4.6.1 Single processor example

For verification of the algorithm and implementation, a simple one dimensional example is examined using the $mE - I$ scheme compared against the Newmark solution for both a linear and a nonlinear system. The benchmark problem is a one dimensional structure of five truss elements with one degree of freedom per node. The structure is fixed at the left end. At the free end a forcing function $R(t) = (1 - \pi^2 f_0^2 t^2) e^{-\pi^2 f_0^2 t^2}$ is applied for $f_0 = 0.4$ Hz. The duration of the loading is 601 steps with a time increment of $\Delta t = 0.01$ sec, and past 6 sec vibrates freely. The properties of the elements are shown in Table 4.1.

Table 4.1. Element properties for simple linear verification example

element	length	young's modulus	density	area
1,2,3	10.0	10^4	1.0	1.0
4,5	30.0	10^6	1.0	1.0

The first analyzes the five truss structure for the linear material case. In a second step the material of elements 4 and 5 is switched to an elastic perfectly plastic material with the same as before Young's modulus and a yield strain $\epsilon_y = 10^{-6}$. The results from both analyses are presented in Fig. 4.7 and Fig. 4.8. The results present very good agreement and thus the correctness of the implementations is established. It can be seen from both plots that as m, the integer ratio of the implicit to explicit time step increases the method starts less accurate and unstable. From preliminary studies ratios of up to 50 are recommended, but this is also a function of the complexity of the problem.

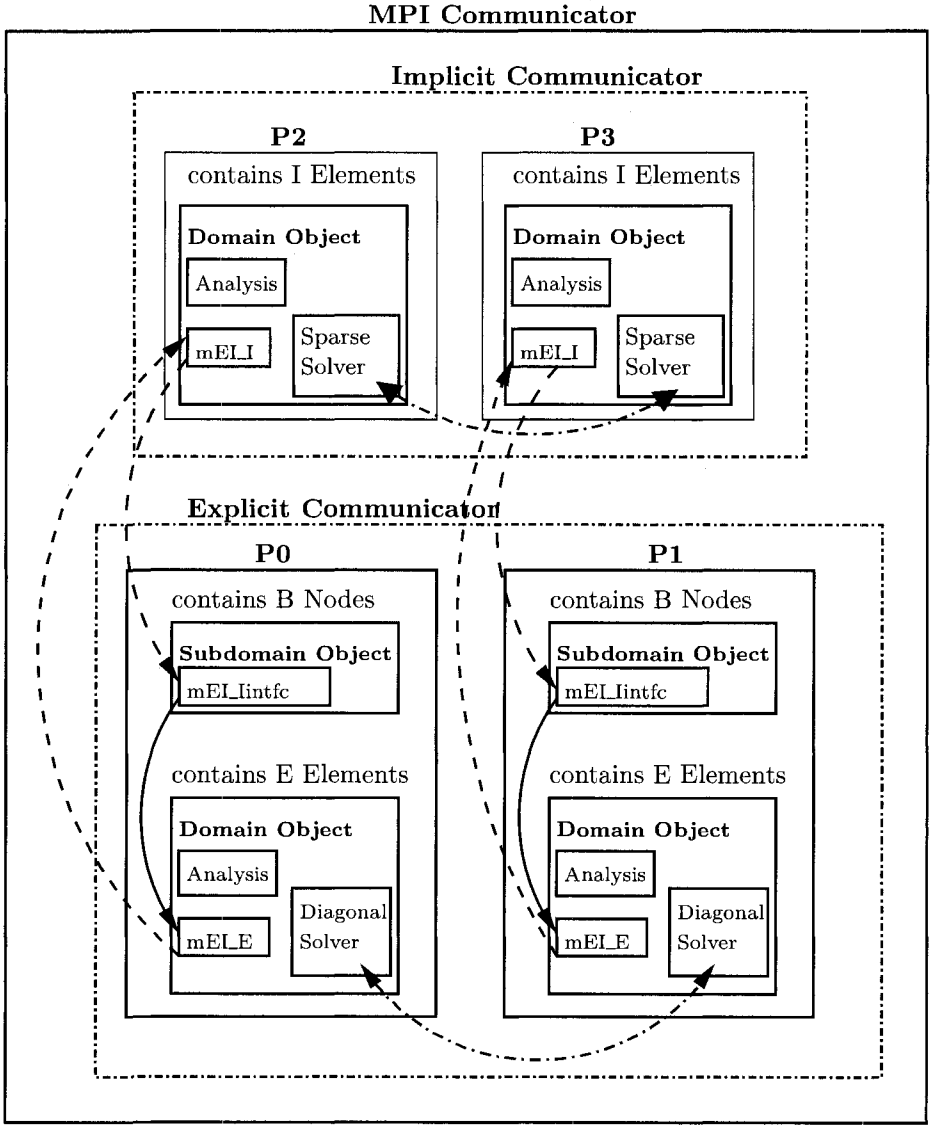

Figure 4.6. generic example of mEI architecture for 4 PEs

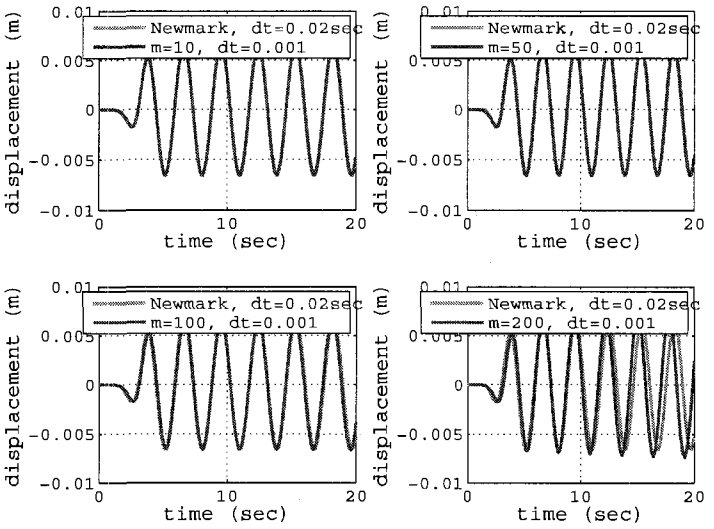

Figure 4.7. Comparison of closed form solution and numerical solution using our custom made code for a simple one dimensional wave propagation problem

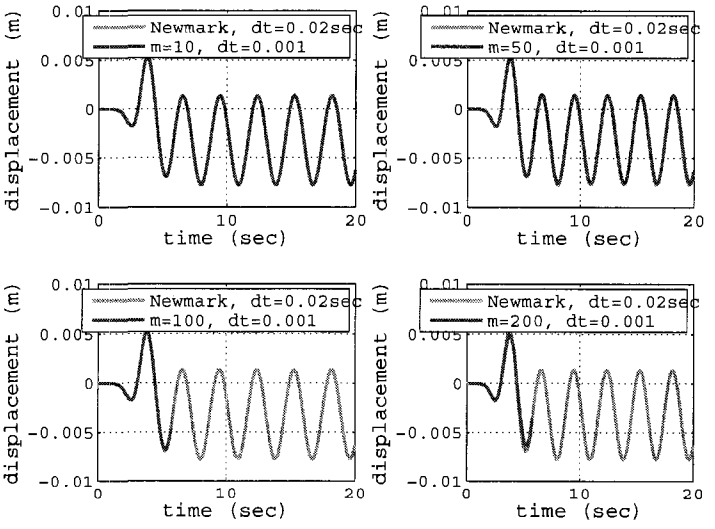

Figure 4.8. Comparison of closed form solution and numerical solution using our custom made code for a simple one dimensional wave propagation problem

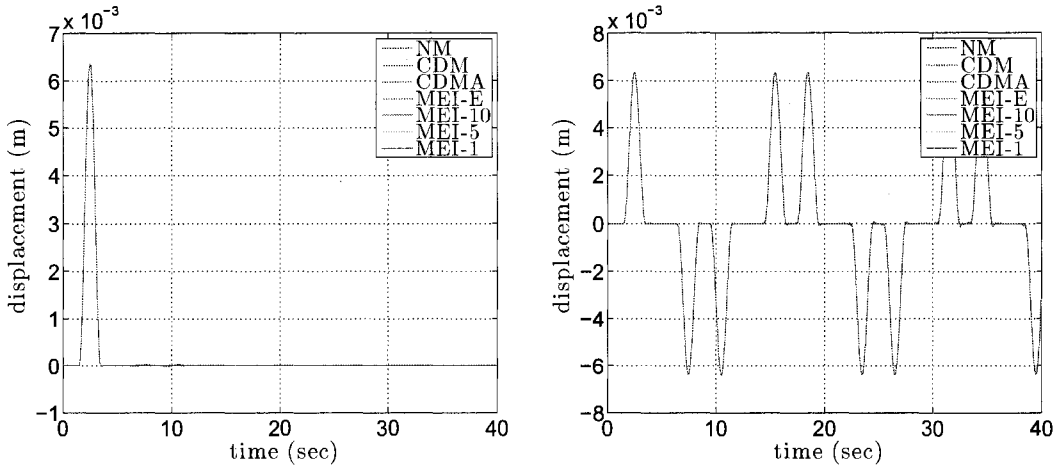

Figure 4.9. One dimensional wave propagation example verification of mixed integration scheme

4.6.2 One dimensional wave propagation verification example

This section contains the same example that was presented in Section 3.1. The domain is analyzed using Newmark's method, the two variants of central difference and the explicit Newmark based method presented in Chapter 2 and finally the domain is also analyzed using mixed integration having half of the 400 elements explicit and the other half implicit. The problem is analyzed for both an absorbing and a fixed boundary condition in the right end. The results are shown in Fig. 4.9.

4.6.3 DRM regular soil verification example

The last test presented is using the entire set of developed code, the solver, soe, dof numberer, integrator classes and element. First the original soil profile presented in Section 3.2 is analyzed using a coarse 20 m element mesh. The center node displacements are recorded. In a second analysis the 4 FLBrick elements around the centernode are analyzed implicitly using mixed integration. The remaining domain is analyzed explicitly. The comparisons present full agreement and are shown in Fig. 4.10.

Several verification tests that assess the correctness of the implementations of the various components of mixed integration have been presented.

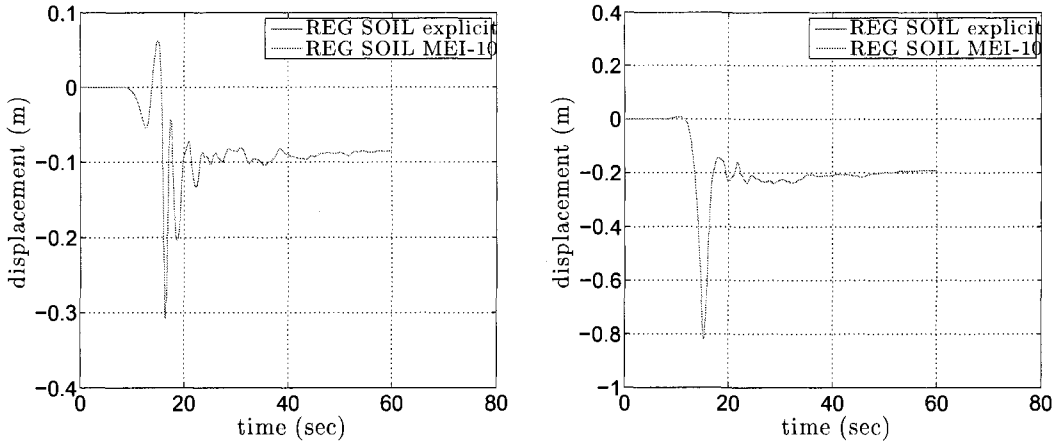

Figure 4.10. Three dimensional wave propagation example verification of mixed integration scheme

Chapter 5

Soil-foundation-structure interaction analysis of urban regions under near-fault excitation

This chapter presents high fidelity, three-dimensional, soi-foundation-structure interaction (SFSI) analyses, using the DRM and simulated input wavefield the 7.1 Mw earthquake from the Puente Hills fault. The structural models used are simplified 6 degree-of-freedom, lumped mass oscillators, with an embedded foundation. The structural characteristics are selected such that they are consistent with the frequency contents of the input, the low frequency simulations on the soft soil described in Section 3.2.5. The goals of this study are to examine the inertial interaction effects and building to building interaction.

5.1 Seismic performance of buildings in a region

Most structural performance simulations to date, are either fixed base analyses or if they account for SFSI they resort to the frequency domain analysis or in the time domain at an individual structure state. The problem of three-dimensional SFSI at a larger regional scale, including local site effects, building to building interaction and with three-dimensional input as opposed to vertically propagating shear waves originating from bedrock upon deconvolution has not been attempted yet.

Park et al. (2004) examined the distribution of damage and inelastic response of building frames for two idealized fault scenarios. In particular it examined the effect of the building location and orientation on the structural damage as well as its vertical distribution over the height of the building. The results were also examined under a seismic code provision perspective and it was concluded that standard design procedures for near-fault ground motions do not prevent soft story formations in the forward directivity zone for tall buildings. Olsen et al. (2008) studied the performance of long-period buildings (twenty story welded moment resisting nonlinear frames) under various earthquake scenarios. The study used ground motions with very high peak ground velocities, in the order of 3 m/sec. It examined differences in the performance of various assumptions about structural design, such as ductile welds versus brittle welds in the frames, with not surprising results. Significant collapse percentiles were shown for the San Francisco Bay Area based on the simulations with the very strong ground motion. The study also examined base isolated buildings.

To examine SFSI of a large bridge, Elgamal et al. (2008) used three-dimensional finite elements with implicit time integration to analyze a long-span bridge with pile and shaft foundations, including a liquefiable soil region. The seismic input was vertically propagating shear waves to analyze a nonlinear model of the bridge structure. The authors identified the need for using more insightful type of inputs such as the one provided by the DRM. Also the implicit integration requires a factorization of the resulting linear form

and this quickly leads to saturation of the computational capabilities, at least when direct solvers are employed.

Detailed discussions and literature review about the characteristics of near-fault ground motion, and the fixed base structural response subjected to it can be found in Park (2004). A comparison of the response of SDOF systems to recorded near and far-fault wave fields was presented in Chopra and Chintanapakdee (2001). The authors concluded that for many near-fault ground motions, the fault normal component imposes higher deformation and strength demands compared to the fault parallel. Also for the same ductility factor the strength demand in the acceleration sensitive period was higher for near-fault motions versus far-fault motions.

5.2 Analysis of representative simplified systems

In order to perform a preliminary study of the structural response subjected to the near-fault wave fields using the DRM, representative systems need to be selected, designed and analyzed. For the purposes of this work structural systems will be represented using simplified models which capture fairly accurately the roof drift and first vibration mode behavior. The focus of this study is on global response. The representative simplified system used is shown in Fig. 5.1. A three-dimensional beam column element, with two nodes and six degrees-of-freedom per node is used (Neuenhofer and Filippou, 1998). The beam element is connected to a rigid column of height equal to the foundation depth. The rigid column's nodal displacements are constraint to be equal to those of the surrounding brick elements that model the foundation. The structural mass m, as a function of the structure's height H, width b and material mass density ρ is calculated according to:

$$m = 0.08Hb^2\rho$$

The system's height is chosen to be $\frac{2}{3}H$ where H is the height of the building. This choice is done in order to represent the first effective modal height for the system. The

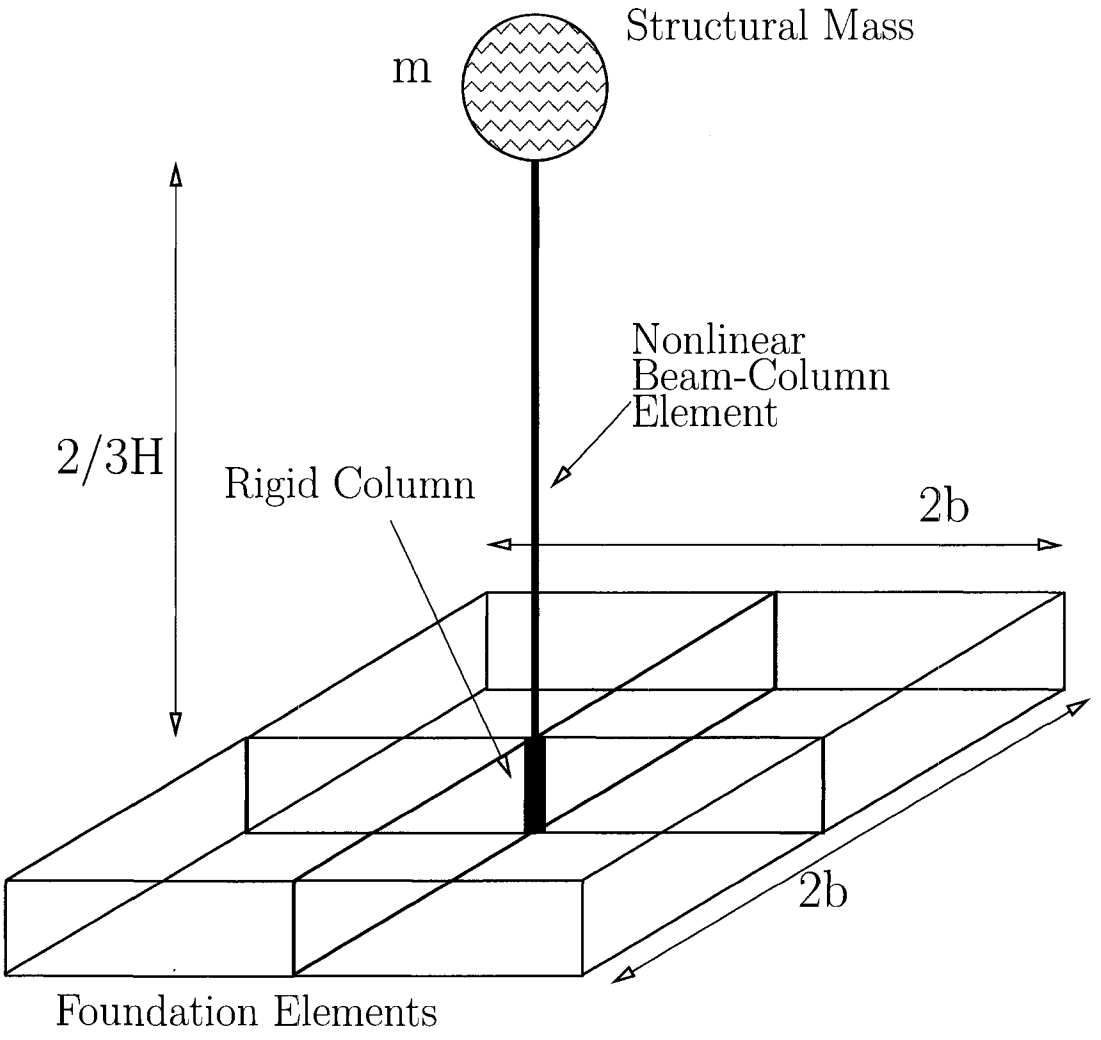

Figure 5.1. Representative SDOF with foundation

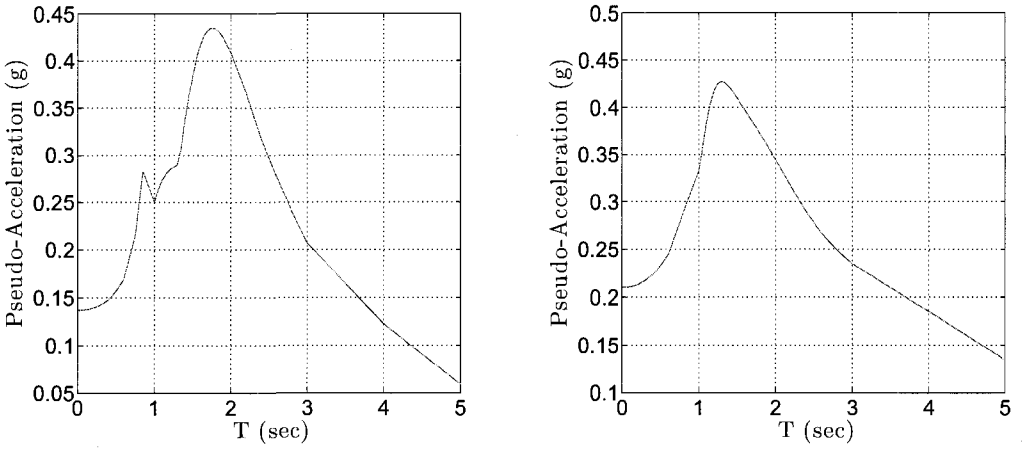

Figure 5.2. X and Y component of 5% damped pseudo acceleration spectrum at top surface centernode

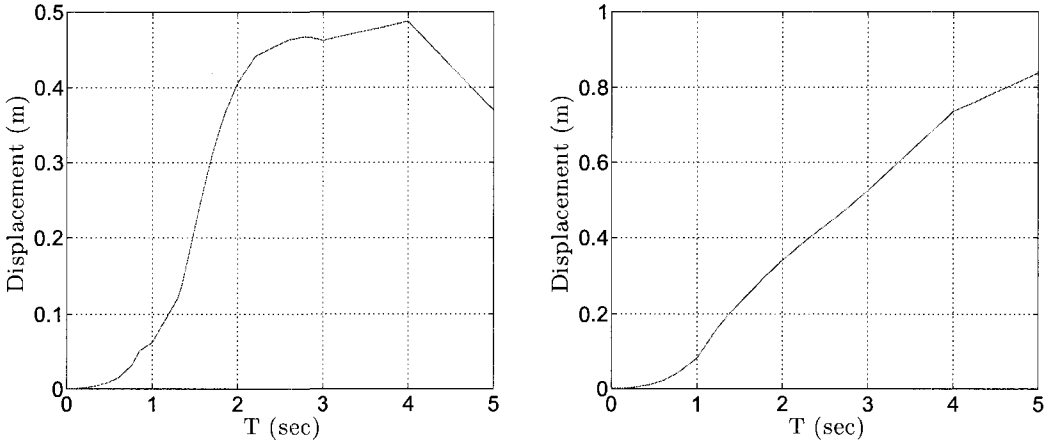

Figure 5.3. X and Y component of 5% damped displacement spectrum at top surface centernode

definition of the latter is given in Chopra (2001). The selection of properties and design of the representative simplified systems is based on the seismic demands at the top surface centernode wave field for the soft soil case. The pseudo-acceleration and displacement spectra corresponding to the center node are shown in Fig. 5.2 and Fig. 5.3 respectively. Given that the input wavefield contains frequencies up to 0.5 Hz the systems selected need to have a vibration period of at least 2 sec. Two representative periods are selected: $T = 2$ sec and $T = 4$ sec. The first for it is close to the peaks of the pseudo-acceleration spectrum and the second one because it is located at the peak of the displacement spectrum for the X direction. For these two categories of systems various designs are selected. The design

procedure is strength based, meaning that the yield reduction factor R_y is varied. The definition of R_y is:

$$R_y = \frac{f_o}{f_y} = \frac{u_o}{u_y}$$

The quantities with y subscript represent yield quantities and the quantities with o subscript represent peak elastic response quantities. For a linear elastic system $R_y = 1$ and as R_y increases beyond one the system experiences increased yielding since the maximum elastic force it can withstand is a reduced fraction of its elastic base shear. This design refers to an idealized elastoplastic system. For the purposes of this study the material law selected is a bilinear hardening material with ratio of post yield to elastic stiffness $\alpha = 0.001$. The choice of this material law over an elastic-perfectly-plastic is for numerical convergence reasons. The foundation elements are modeled using FLBrick elements whose density is adjusted to maintain a structural to foundation mass ratio of five: $\frac{m}{m_f} = 5$.

Finally other quantities that will be used throughout are: ductility which is defined as:

$$\mu = \frac{u_{max}}{u_o}$$

where quantities with max subscript indicate the maximum for an inelastic analysis; a quantity related to simplified soil-foundation-structure interaction σ that relates the soil to the structural stiffness:

$$\sigma = \frac{V_S T}{H}$$

and the seismic coefficient C_{max} which is defined as the ratio of the maximum base shear the system experience over its weight:

$$C_{max} = \frac{f_{max}}{W}$$

5.2.1 Two second period systems

Three different, two second period systems are examined. Table 5.1 contains the properties of the $T = 2$ sec systems. Fig. 5.4 shows the ground displacement comparison

between free field and SFSI due to T2M4. Fig. 5.5 through Fig. 5.6 show results for T2M1. Fig. 5.7 through Fig. 5.8 show results for T2M3. Fig. 5.9 through Fig. 5.10 contains certain plots of interest for the T2M4 system. The comparison of the ground motion at the top surface center node due to free field versus the wave field containing the SFSI effects shows no significant alteration. This essentially means very small inertial interaction effects, which is expected given the foundation width which is $b = 20$ m and the wavelengths involved that are at least 80 m and thus the base slab averaging is not significant. The comparison for the ground motion due to SFSI from T2M1 and T2M3 is similar to the T2M4 case.

Fig. 5.5 shows period lengthening $\frac{\tilde{T}}{T} \approx 1.11 - 1.16$ for T2M1. Fig. 5.7 shows period lengthening $\frac{\tilde{T}}{T} \approx 1.07 - 1.17$ for T2M3. Fig. 5.9 shows period lengthening $\frac{\tilde{T}}{T} \approx 1.1 - 1.25$ for T2M4 where $\tilde{T}$ is the period of the flexible base system. These values are computed by looking at the first four peaks of the deformation plots. As expected the period lengthening effect is more noticeable for the system that yields more, the T2M4. The concept of period for a yielding system with flexible base is hard to define concretely. The reported numbers refer to the ratios of time required between successive peaks with the same sign of velocity before them, i.e. two peaks on the positive or negative displacement. These ratios if compared to the solutions by (Bielak, 1975; Veletsos and Nair, 1975) which are also reported in Stewart et al. (1999a) indicate that the simplified closed form solutions can be overestimating some of these effects. For example for $\frac{1}{\sigma} = \frac{H}{V_S T} = 0.25$ as is the case for T2M3 and T2M4, for an aspect ratio $\frac{H}{r} = 5$, Bielak (1975); Veletsos and Nair (1975) estimate $\frac{\tilde{T}}{T}$ to ≈ 1.35 and ≈ 1.18 respectively. No significant alterations are observed as far as the damping for the $T = 2$ sec systems.

The effect of the yielding material law results in the structure experiencing permanent deformations. This is reflected in the design via the increasing R_y factor. For the T2M1 there is no significant permanent offset as expected. Interestingly while in the T2M4 case the permanent offsets are reduced, for the T2M3 case the permanent deformations are

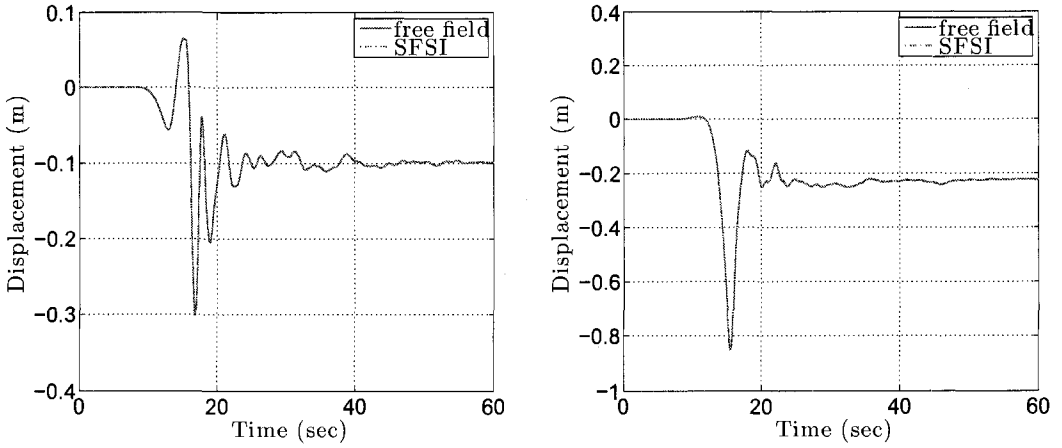

Figure 5.4. Comparison of top surface centernode displacements for free field versus kinematic and inertial interaction due to T2M4, X(left) and Y(right)

increased in the SFSI case versus the fixed base case. This shows the plethora of studies and results that can be generated. Great variability in results is observed for a single system for varying the R_y strength design parameter. In the T2M4 system, in the Y direction an increased structural deformation occurs and in all other cases the structural deformations are reduced. Table 5.2 contains the ductility and seismic coefficient demands for the $T = 2$ sec systems analyzed using SFSI.

Table 5.1. $T = 2$ sec properties

	H (m)	b (m)	H^*	ζ (%)	u_o (m)	R_y	$\frac{m}{m_f}$	α	$\frac{1}{\sigma}$
T2M1	80	20	$\frac{2}{3}H$	5	0.448	1	5	0.001	0.25
T2M3	80	20	$\frac{2}{3}H$	5	0.448	3	5	0.001	0.25
T2M4	80	20	$\frac{2}{3}H$	5	0.448	4	5	0.001	0.25

Table 5.2. $T = 2$ sec results

	u_{xmax} (m)	$u_{y_{max}}$ (m)	u_{max} (m)	C_{max}	μ
T2M1	0.351	0.283	0.3976	0.39	0.888
T2M3	0.23	0.37	0.396	0.152	2.651
T2M4	0.181	0.4435	0.4778	0.119	4.266

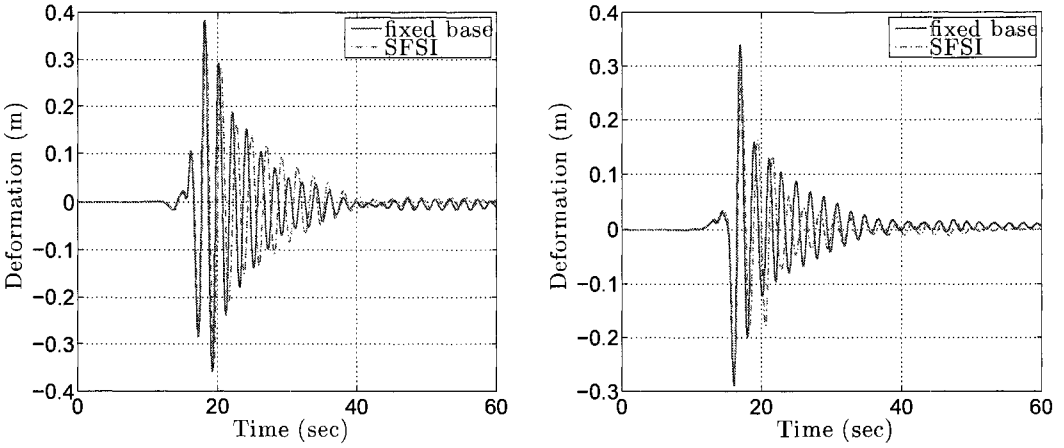

Figure 5.5. Fixed base versus SFSI structural deformations for T2M1, X(left) and Y(right)

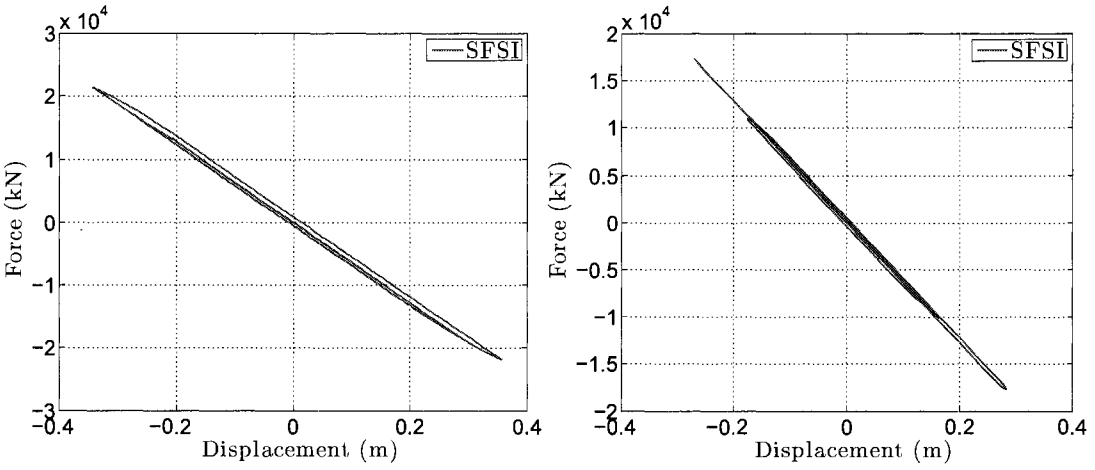

Figure 5.6. SFSI force-deformations for T2M1 X(left) and Y(right)

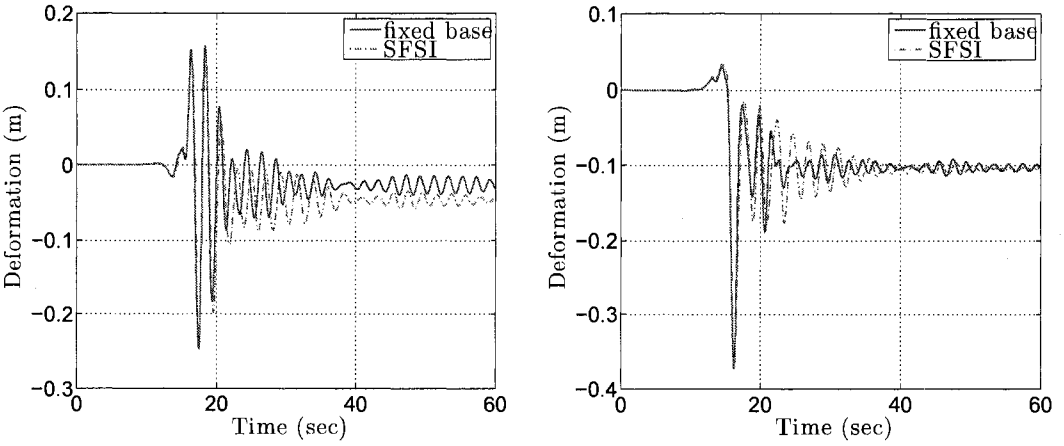

Figure 5.7. Fixed base versus SFSI structural deformations for T2M3, X(left) and Y(right)

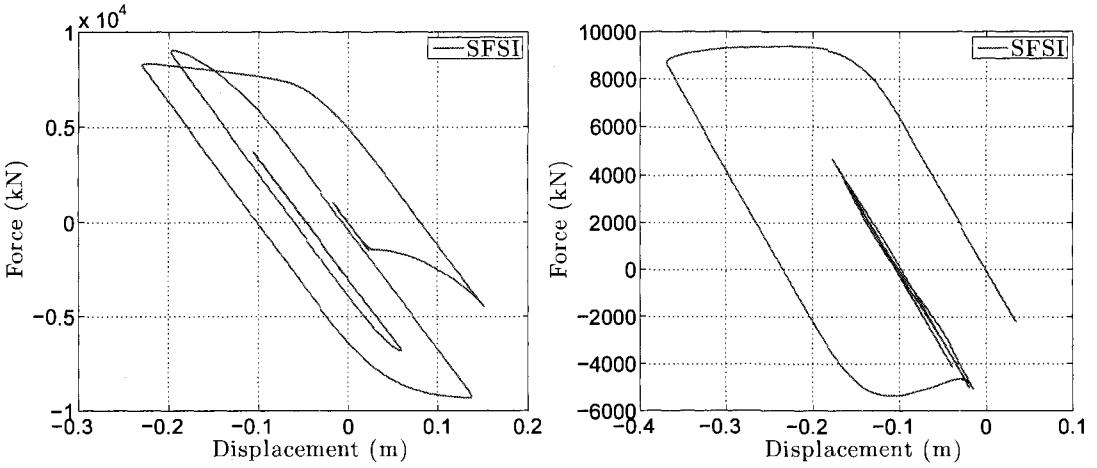

Figure 5.8. SFSI force-deformations for T2M3, X(left) and Y(right)

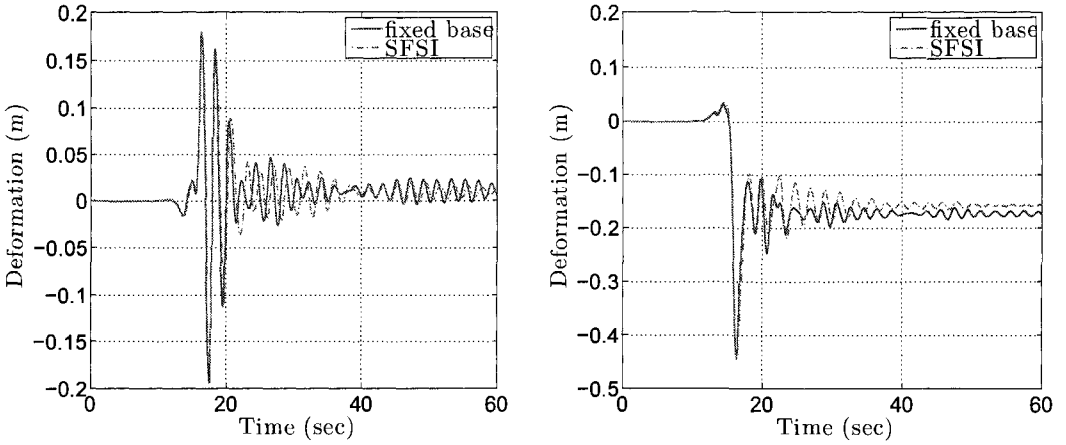

Figure 5.9. Fixed base versus SFSI structural deformations for T2M4, X(left) and Y(right)

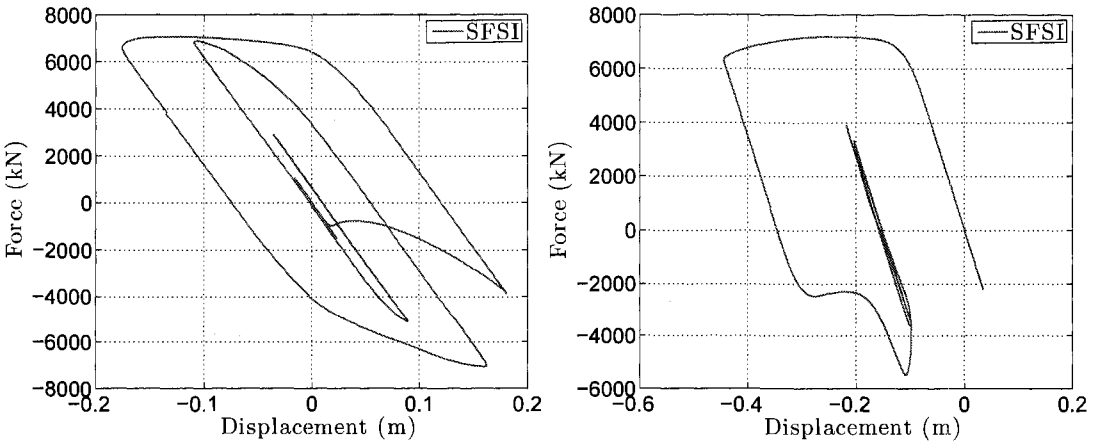

Figure 5.10. SFSI force-deformations for T2M4, X(left) and Y(right)

5.2.2 Four second period systems

Table 5.3 contains the properties of the $T = 4$ sec systems. Fig. 5.11 shows the ground displacement comparison between free field and SFSI due to T4M2. Fig. 5.12 through Fig. 5.13 contains certain plots of interest for the T4M1 system, and Fig. 5.14 through Fig. 5.15 for the T4M2. As expected for the $T = 4$ sec systems the ground motion alteration due to the wider foundation and increased mass is more noticeable compared to the lighter systems with narrower foundations. Base slab averaging is observable in this case.

As expected for the 4 sec systems the period lengthening is higher compared to the 2 sec ones since the former have a higher $\frac{1}{\sigma}$. $\frac{\tilde{T}}{T} \approx 1.3$ both for the T4M1 and the T4M2. Again the simplified closed form solutions appear to be overestimating the period lengthening effect to about 1.5. An effect that was not present in the $T = 2$ sec and is observed in the $T = 4$ sec case is a reduction in the effective damping. The damping for both systems is reduced, mainly in the X direction. The effective damping $\tilde{\zeta}$ from Eq. (1.2) has two components: one due to the radiation and hysteretic damping in the foundation and one due to viscous damping in the structure respectively (Stewart et al., 1999a). The first term is estimated in the literature (Veletsos and Nair, 1975) to be about 2-3 %. The second term comes to about 2.3 % and thus the reduction observed verifies the simplified solutions (Stewart et al., 1999a). It appears that the results in Bielak (1975) overestimated $\tilde{\zeta}_0$ and that Veletsos and Nair (1975) provided more realistic estimates for the effective damping. In order to better understand the damping ratio reduction, the contribution of rocking to the structural deformation is examined in Fig. 5.16 and Fig. 5.17. Rocking is significant for the $T = 4$ sec systems and also appears to have a reduced damping factor compared to the rest of the structural deformation (resulting from the displacement difference of the top versus the base).

The permanent offset is reduced for T4M2 under SFSI and in both systems the structural deformation demands are reduced compared to the fixed base results. Table

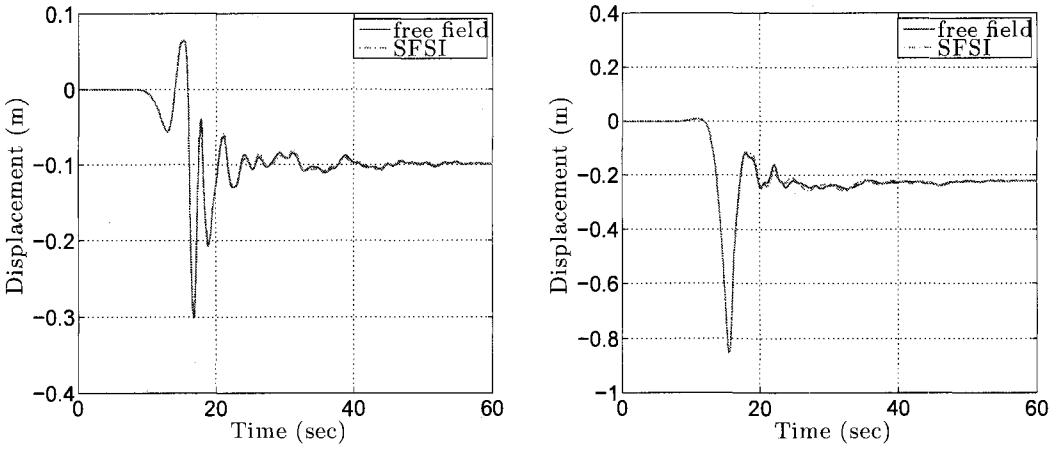

Figure 5.11. Comparison of top surface centernode displacements for free field versus kinematic and inertial interaction due to T4M1, X(left) and Y(right)

5.4 contains the seismic coefficient and ductility demands for the $T = 4$ sec systems when analyzed using SFSI.

Table 5.3. $T = 4$ sec properties

	H (m)	b (m)	H^*	ζ (%)	u_o (m)	R_y	$\frac{m}{m_f}$	α	$\frac{1}{\sigma}$
T4M1	180	40	$\frac{2}{3}H$	5	0.751	1	5	0.001	0.281
T4M2	180	40	$\frac{2}{3}H$	5	0.751	2	5	0.001	0.281

Table 5.4. $T = 4$ sec results

	u_{xmax} (m)	$u_{y_{max}}$ (m)	u_{max} (m)	C_{max}	μ
T4M1	0.24	0.509	0.521	1.184	0694
T4M2	0.276	0.576	0.6187	0.866	1.649

5.3 Response of two second vibration period system for two different meshes

Regional simulations that involve more than one structural model require higher resolution meshes in order to resolve the presence of higher frequencies due to the interaction between the foundations of neighboring buildings. This section discusses the differences in the soil response including SFSI due to T2M3 and the differences in the structural deformations including SFSI of the system T2M3 when simulated with a 10 m versus a 5 m

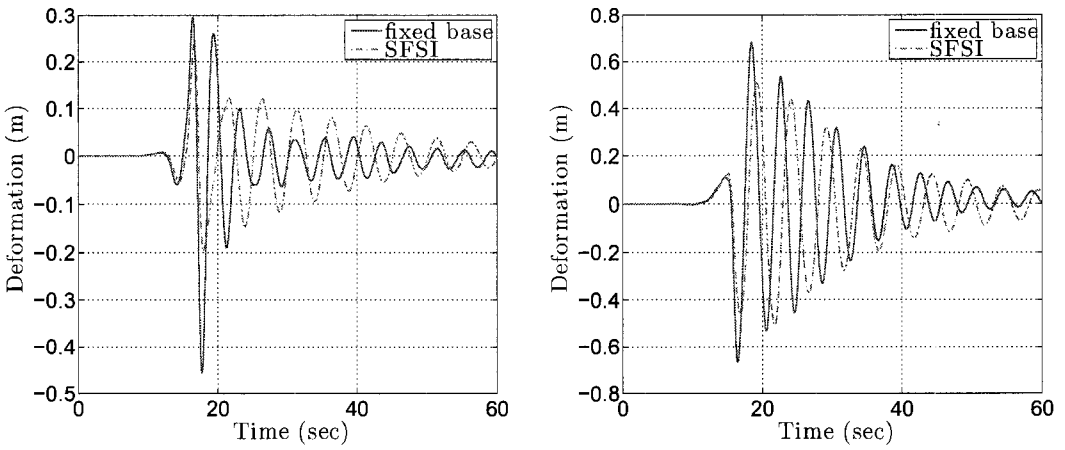

Figure 5.12. Fixed base versus SFSI structural deformations for T4M1, X(left) and Y(right)

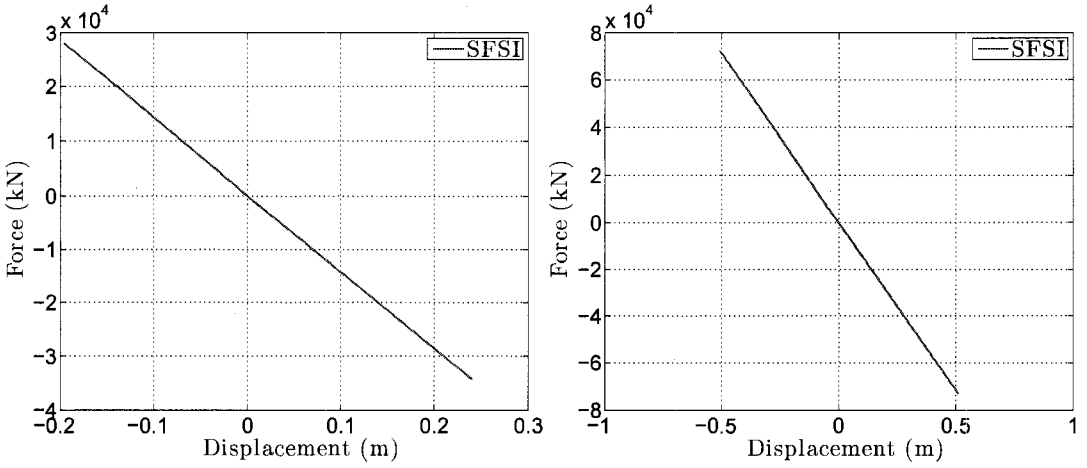

Figure 5.13. SFSI force-deformations for T4M1, X(left) and Y(right)

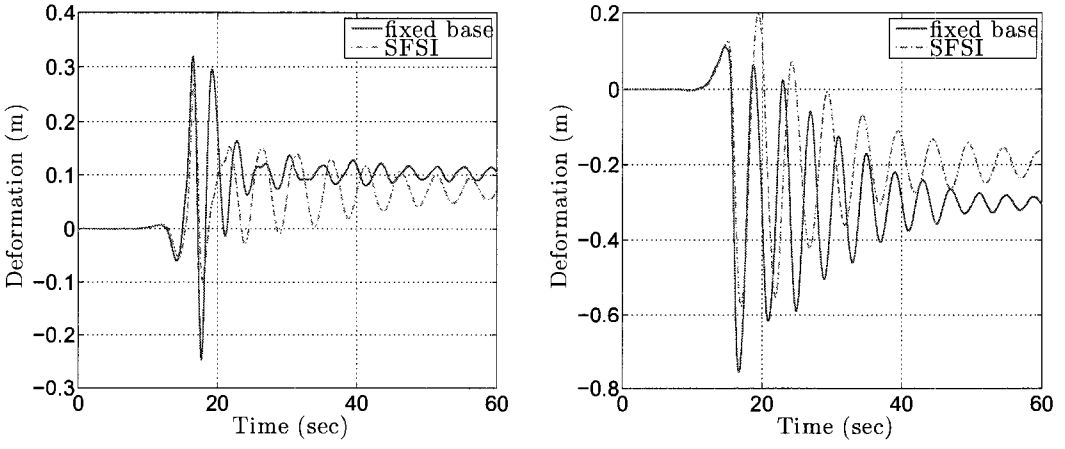

Figure 5.14. Fixed base versus SFSI structural deformations for T4M2, X(left) and Y(right)

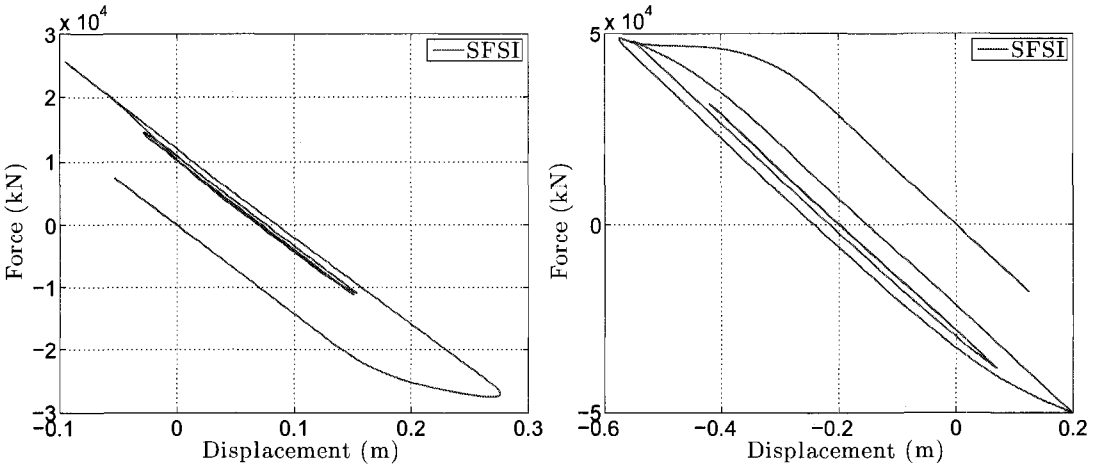

Figure 5.15. SFSI force-deformations for T4M2, X(left) and Y(right)

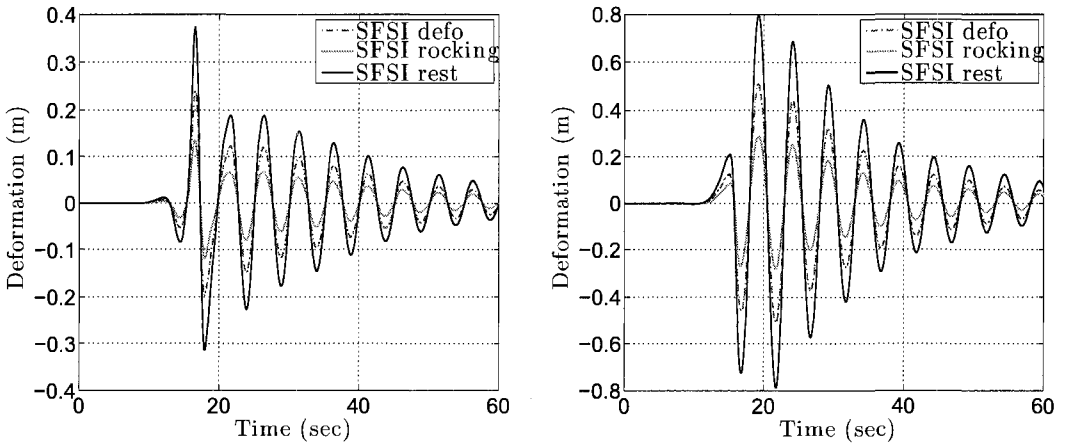

Figure 5.16. Contribution of rocking to structural deformation for T4M1, X(left) and Y(right)

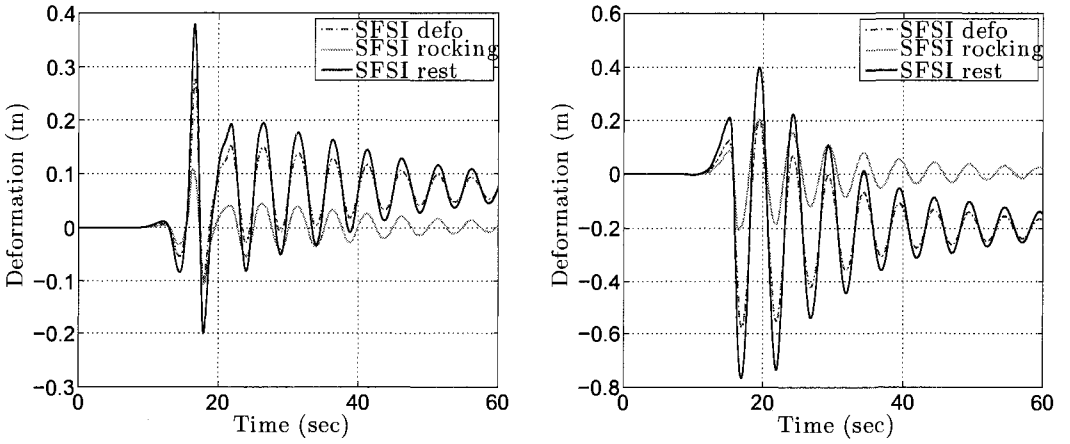

Figure 5.17. Contribution of rocking to structural deformation for T4M2, X(left) and Y(right)

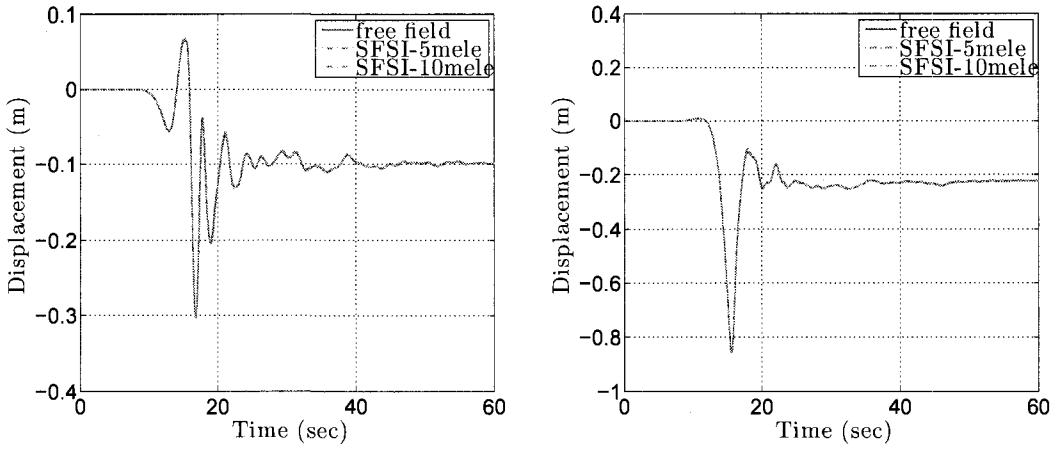

Figure 5.18. Comparison of the effect of the mesh size on the soil displacement at the centernode due to the T2M3 system for a 5 m element mesh versus a 10 m element mesh, X(left) and Y(right)

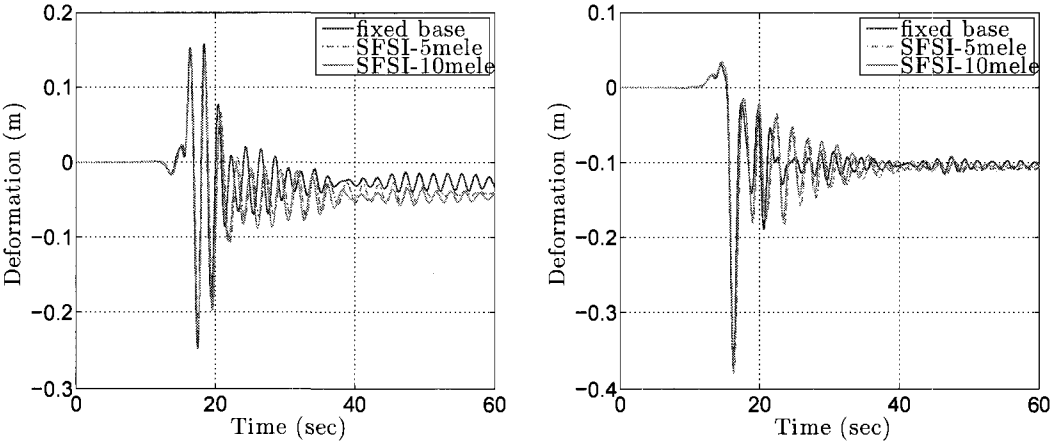

Figure 5.19. Comparison of the effect of the mesh size on the structural deformation of the T2M3 system for a 5 m element mesh versus a 10 m element mesh, X(left) and Y(right)

element mesh. The soil response due to SFSI from the T2M3 structure is plotted in Fig. 5.18 and the structural deformations of T2M3 due to SFSI are plotted in Fig. 5.19. Both figures show that the refined mesh has negligible effect on the simulated response. Peak responses for the structural system are presented in Table 5.5, showing that the response does not differ significantly with the two meshes. In the remaining of this chapter, the T2M3 responses computed for the multiple building analyses are compared against the response of the T2M3 system for the 5 m mesh.

Table 5.5. T2M3 results for 10 m and 5 m mesh

	u_{xmax} (m)	u_{ymax} (m)	u_{max} (m)	C_{max}	μ
T2M3 (10 m mesh)	0.23	0.37	0.396	0.152	2.651
T2M3 (5 m mesh)	0.223	0.379	0.405	0.152	2.713

5.4 Simulation with five, spatially distributed structural systems

This section presents the coupled analysis of five neighboring buildings, each with a two second vibration period and strength reduction factor of three (T2M3). The buildings are spaced at 100 m in two directions. The clear separation between the foundations is 80 m. The soil conditions are the same soft soil profile previously discussed and the input is the low frequency wavefield discussed in Chapter 3. The mesh size is a uniform 5 m mesh for a total of 9.36 million FLBrick elements analyzed on 1024 processors at Abe in NCSA (2008).

Fig. 5.20 through Fig. 5.22 show the spatial distribution of the foundation displacements, velocities and accelerations respectively. Overall, the variability of the foundation motion is small, but is most noticeable in the velocity histories because of the predominant period of free-field ground motion. The structural deformations of the T2M3 system due to SFSI are shown in Fig. 5.23. The centernode system's response along the X and Y axes differs little compared to the data in Table 5.2, the maximum displacement on the X direction changes from 0.23 m to 0.223 m and for the Y direction from 0.37 m to 0.378 m. The structural deformations between the T2M3 at the center of the multiple building analysis and the case of one system are very similar. The ductility demand for the buildings, plotted in Fig. 5.24, is slightly increased for the centernode system ($X_{centernode}, Y_{centernode}$) = (500 m, 250 m), when compared with a value of 2.651 for the single T2M3 case. The seismic coefficient shown in Fig. 5.24 for the centernode system is unchanged compared again with the single T2M3 case that had a value of 0.152 as shown in Table 5.2. From Fig. 5.24 that the highest demands are for the system with the lowest Y coordinate. The

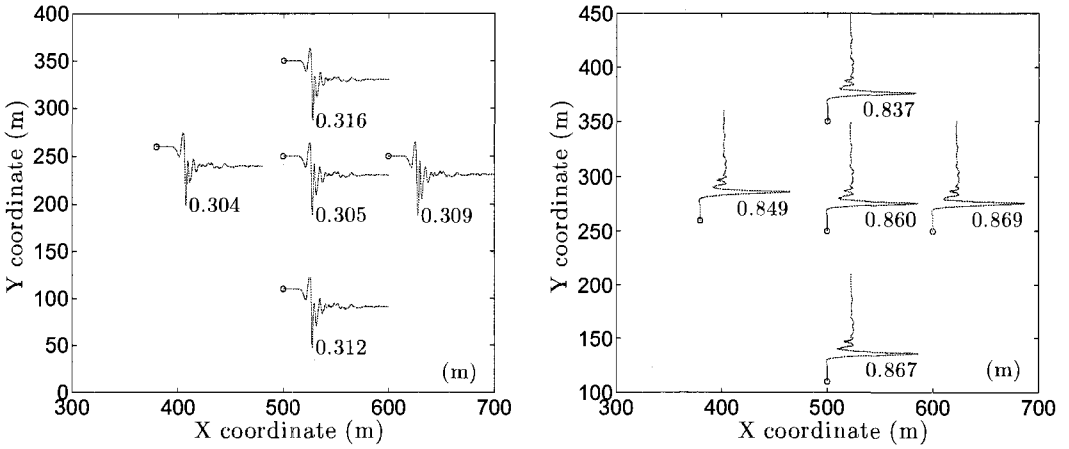

Figure 5.20. Spatial variation in foundation displacement history, including SFSI due to T2M3, X-component of displacement (left) and Y-component of displacement (right). Circles indicate the location of the foundation and the structure with respect to Fig. 3.3. The value of peak displacement (m) is indicated in the plots

results indicate that the 80 m clear distance at which the structural systems are located from each other is sufficiently large to limit building to building interaction effects for this range of parameters and the low-frequency ground motion.

5.5 Simulation with six, spatially concentrated structural systems

This section presents the coupled analysis of six neighboring buildings, each with a two second vibration period and strength reduction factor of three (T2M3). The buildings are spaced at $20m$ in two directions. The clear separation between the foundations is 10 m. The soil profile and the mesh are identical to the previous section. The focus is on the response of the centernode system and how it is affected by the neighboring structures. The spatial variation of the foundation displacements, velocities and accelerations is shown in Fig. 5.25 through Fig. 5.27.

The structural deformation time histories for the centernode T2M3 system and their

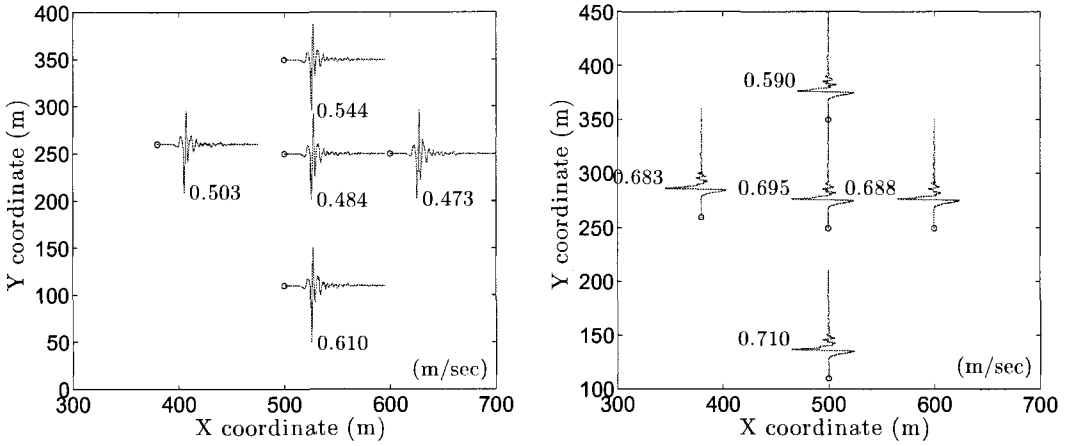

Figure 5.21. Spatial variation in foundation velocity history, including SFSI due to T2M3, X-component of velocity (left) and Y-component of velocity (right). Circles indicate the location of the foundation and the structure with respect to Fig. 3.3. The value of peak velocity (m/sec) is indicated in the plots

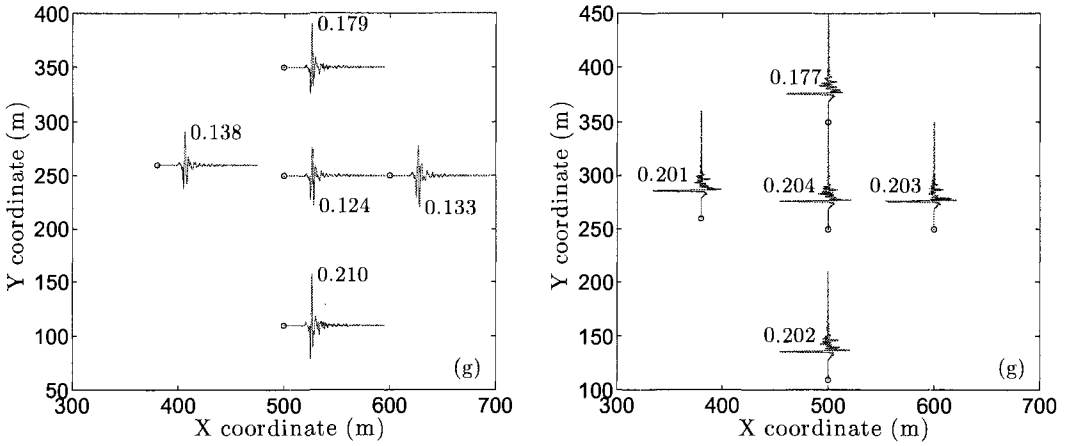

Figure 5.22. Spatial variation in foundation acceleration history, including SFSI due to T2M3, X-component of acceleration (left) and Y-component of acceleration (right). Circles indicate the location of the foundation and the structure with respect to Fig. 3.3. The value of peak acceleration (g) is indicated in the plots

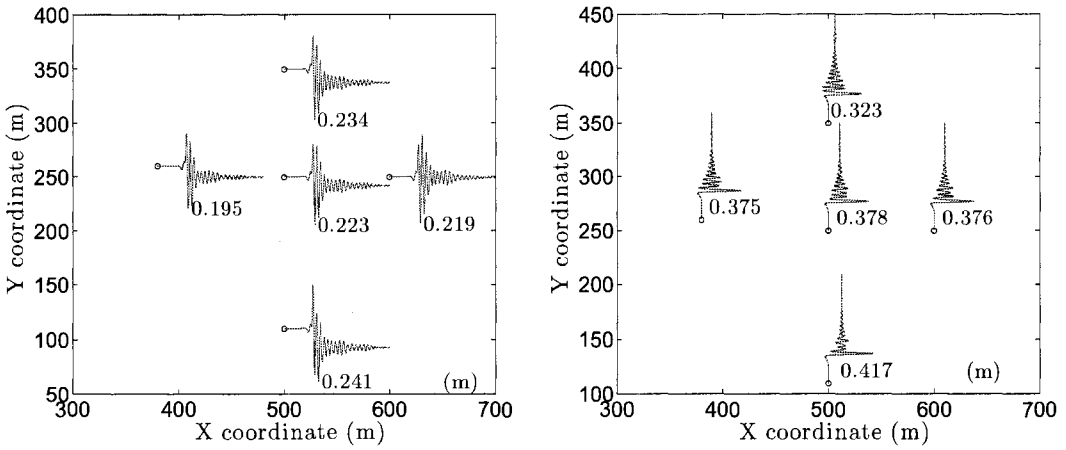

Figure 5.23. Spatial variation in structural deformation history, including SFSI, X-component of structural deformation (left) and Y-component of structural deformation (right). Circles indicate the location of the foundation and the structure with respect to Fig. 3.3. The value of peak structural deformation (m) is indicated in the plots

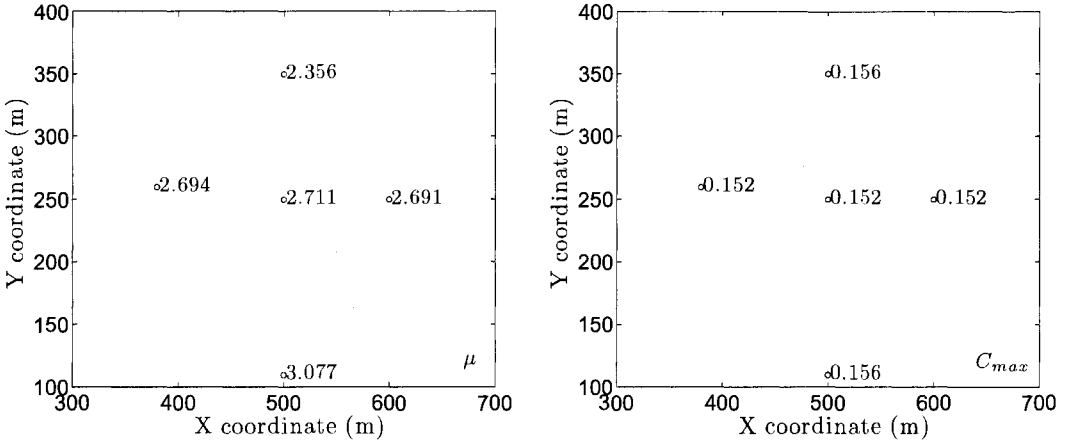

Figure 5.24. Spatial variation of ductility(left) and seismic coefficient (right) of T2M3 including SFSI. Circles indicate the location of the foundation and the structure with respect to Fig. 3.3

comparison with the case when T2M3 is analyzed alone are shown in Fig. 5.28. A small increase in the permanent deformation is observed in the X direction for the six buildings case. The differences appear to be small, which given the close proximity of the structures, indicates that the kinematic interaction between the foundations is small. The mass of the T2M3 (6,400 ton) system is roughly one half of the mass of a soil volume (12,000 ton) equal to its foundation volume (8,000 m^3) and the foundation mass is one fifth of that. Inertial interaction is therefore not significant in this case.

Within this small region almost no spatial variation is observed. The structural deformation time histories are plotted in Fig. 5.30 and the amplitudes are approximately equal to those of the individual T2M3: 0.23 m on the X direction and 0.37 m on the Y direction, with the exception of the system on the lower right corner of the figure. This system's response is the highest of all six in the X direction and the lowest of all six in the Y direction. The peak structural deformation on the X direction, of this system is 50% higher of all the other and the peak structural deformation on the Y direction is roughly 50% lower from the other five systems. This does not closely follow the pattern of the spatial distribution of the foundation displacement or its derivatives. Accelerations show the same trend in terms of the location of the peak values but not with the same percentile differences. The response of the T2M3 at the location of (520 m, 230 m) is examined more closely in Fig. 5.29. A significant increase in the permanent deformation on the X direction is observed, in addition to the increased peak deformation in the same direction. This shows that within a distance of 28 m from the centernode site the structural response is very different and shows why detailed regional analyses are a very useful tool. This diverse response at such a close distance would not be possible to be predicted via means of simplified analyses, or just by looking at the wavefield. The regional simulations of this type incorporates the effects of the fault rupture, wave propagation path, and basin effects through the DRM, and also incorporates: the local site effects, building to building interaction and individual structural and foundation characteristics in one

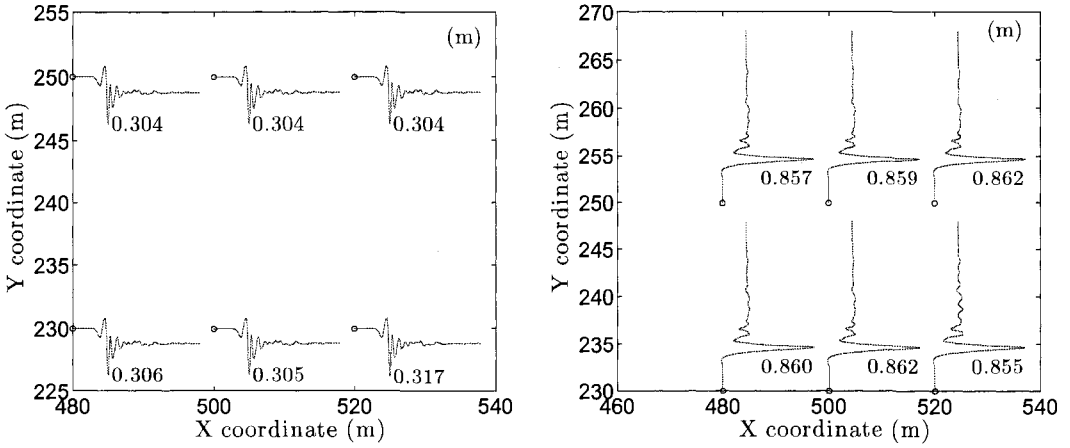

Figure 5.25. Spatial variation in foundation displacement history, including SFSI due to T2M3, X-component of displacement (left) and Y-component of displacement (right). Circles indicate the location of the foundation and the structure with respect to Fig. 3.3. The value of peak displacement (m) is indicated in the plots

analysis. Finally, the ductility and seismic coefficient demands are shown in Fig. 5.31 and also are very close to 2.651 and 0.152 for the individual system respectively.

5.6 Performance and scalability of the regional simulations

The simulations that are presented herein are performed at NCSA (2008) Abe. Details about the architecture, the interconnect and the file system can be found online[1]. These simulations are conducted using the 5 m element mesh, 9.36 million elements on 128 nodes with 8 cores per node, for a total of 1024 cores. The runs are wave propagation only on 1024 PEs, wave propagation only on 1023 PEs and 1 PE with mixed integration with a T2M3 system, 1019 PEs with wave propagation only and 5 PEs with mixed integration each with a T2M3, and finally one run with a large mixed integration problem on one PE and 1023 PEs with wave propagation only. Due to technical reasons the results are presented in two batches: one refers to a software stack containing MVAPICH2 0.9.8 (MVAPICH:, 2008), MKL 9, and Intel compiler 10.0 and the second batch is with MVAPICH2 1.2,

[1]http://www.ncsa.uiuc.edu/UserInfo/Resources/Hardware/Intel64Cluster

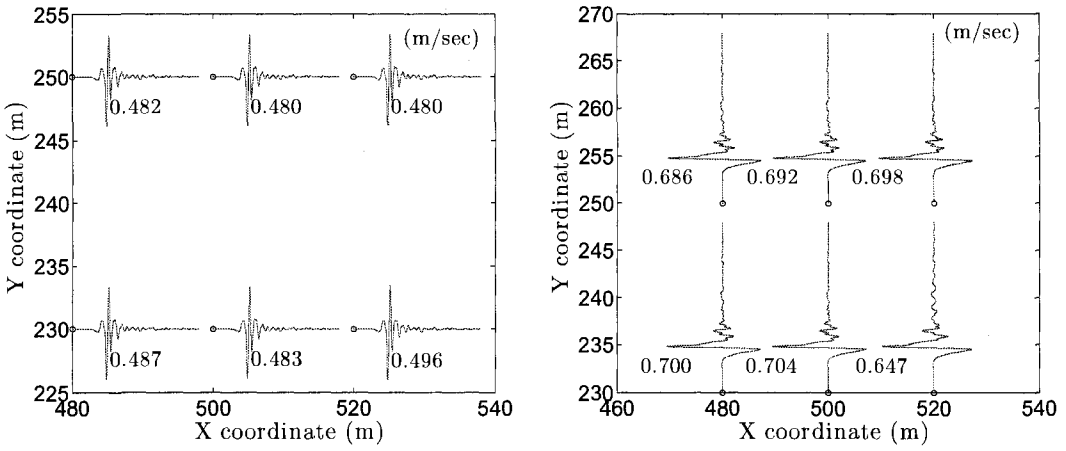

Figure 5.26. Spatial variation in foundation velocity history, including SFSI due to T2M3, X-component of velocity (left) and Y-component of velocity (right). Circles indicate the location of the foundation and the structure with respect to Fig. 3.3. The value of peak velocity (m/sec) is indicated in the plots

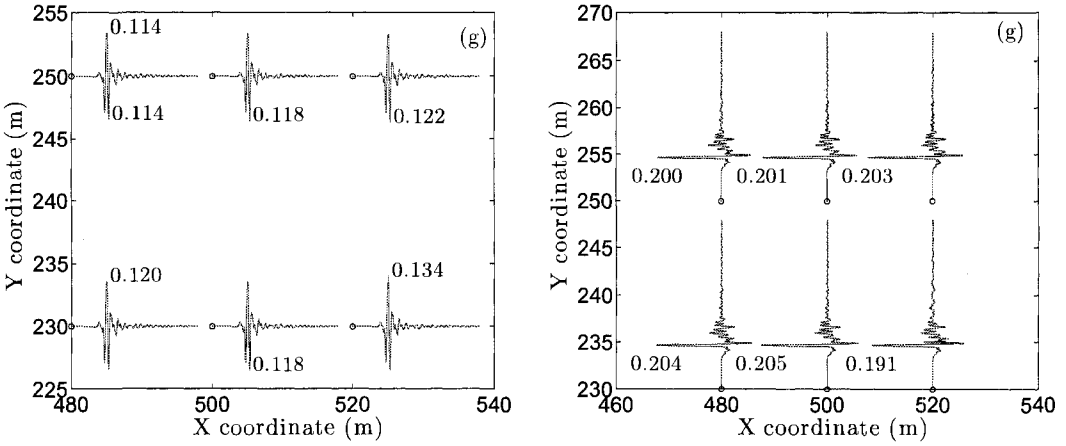

Figure 5.27. Spatial variation in foundation acceleration history, including SFSI due to T2M3, X-component of acceleration (left) and Y-component of acceleration (right). Circles indicate the location of the foundation and the structure with respect to Fig. 3.3. The value of peak acceleration (g) is indicated in the plots

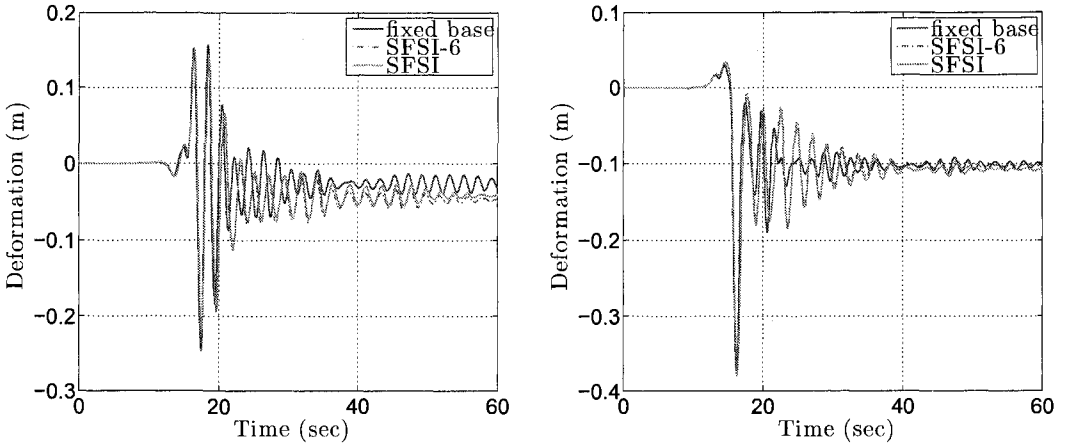

Figure 5.28. Comparison of the structural deformation of T2M3 system at the centernode when analyzed alone and as part of the six building simulation. X (left) and Y (right)

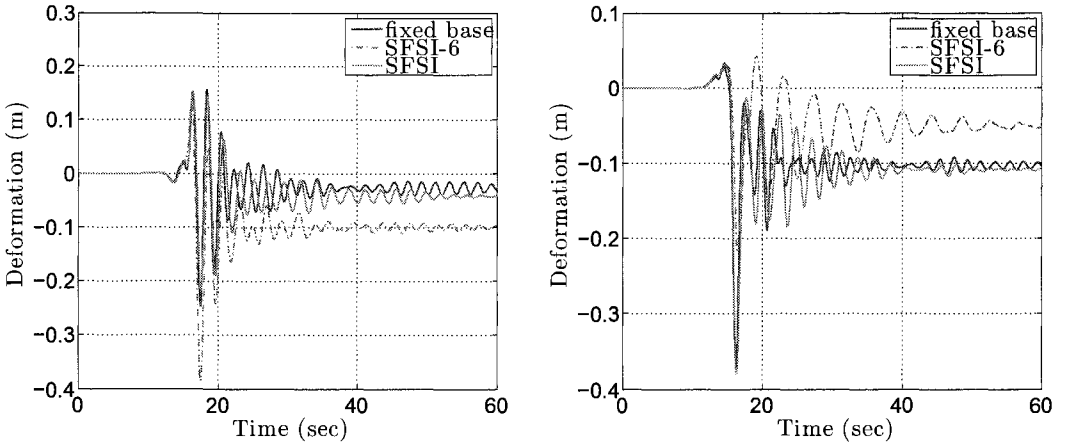

Figure 5.29. Comparison of the structural deformation of T2M3 system at the centernode when analyzed alone against the system at the lower right corner from the six building simulation. X (left) and Y (right)

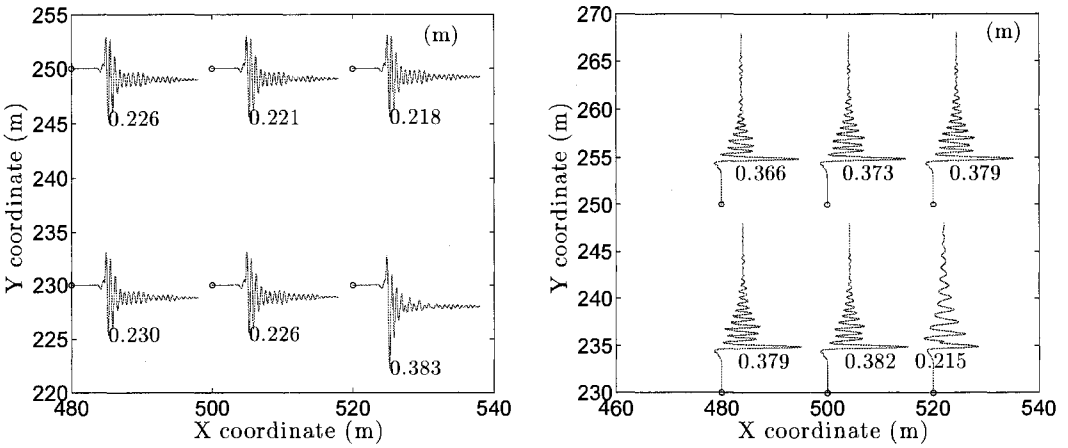

Figure 5.30. Spatial variation in structural deformation history, including SFSI, X-component of structural deformation (left) and Y-component of structural deformation (right). Circles indicate the location of the foundation and the structure with respect to Fig. 3.3. The value of peak structural deformation (m) is indicated in the plots

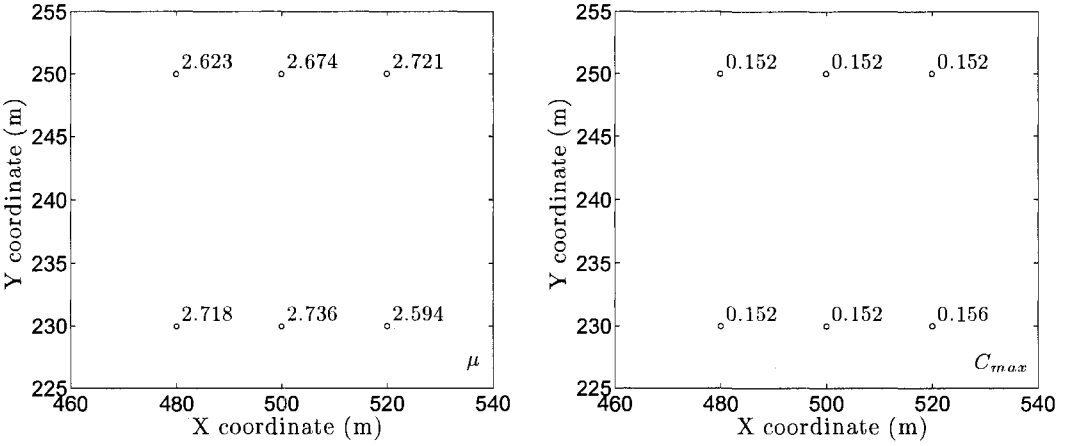

Figure 5.31. Spatial variation of ductility(left) and seismic coefficient (right) of T2M3 including SFSI. Circles indicate the location of the foundation and the structure with respect to Fig. 3.3

MKL 10. and Intel compiler 10.1. The first batch is used for characterizing the scalability of mixed integration with respect to the wave propagation problem only. The second batch compares various analyses that all perform in one or more ranks mixed integration. The conclusions drawn by the first batch can carry to the discussion for the second since the only difference between common types of runs is the software stack. Differences due to network traffic and congestion and task placement on the supercomputer cannot be accounted for since exclusive usage of the machine is not an option. The performance characteristics are based on the PAPI (2008) counters and are collected with PerfSuite (2008).

The first two runs, presented in Table 5.6, compare the effect on the performance of the software, of a small implicit subpartition on one of the ranks to that of a pure wave propagation problem, in which all the PEs solve an explicit problem. The implicit part is relatively small, one 6 degree-of-freedom per node beam-column element with five fiber sections, one elastic beam-column element with six degrees-of-freedom and 32 brick elements for the foundation. While small, this implicit component increases the computational unbalance in the entire communicator. While the rank that owns this implicit domain performs the LU factorization and solution of the system of equations that correspond to it, the neighboring ranks may have to wait in a barrier call longer than the wait time for a pure explicit solution and so do their neighbors and so on. The usage of asynchronous communication in the solver assists in hiding some of this unbalance. The end-to-end time effect is 1.15% which is an encouraging result as far the efficiency of the implementation of the mixed integration and the overhead it adds. It appears that as long as the ratio of sizes of the implicit over the explicit partition in a processor remains small the mixed integration implementation scales fine. Note that the computational complexity of the explicit diagonal solve is linear in the size of the unknowns $O(N)$, while that of the implicit partition is $O(\frac{2}{3}N^3)$ (Demmel, 1997). Also the effect of nonlinear constitutive materials for the structure is hidden by the size of the time step used. The

explicit integration time step is 0.00025 sec and the m of the $mE - I$ is 10, so there are almost no Newton-Raphson iterations required on the implicit subpartition.

Table 5.6. T2M3 simulation performance results using MPVAPICH2 0.9.8, Intel 10.0.026, and MKL 9.1.023

# mixed PEs	1	0
# explicit PEs	1023	1024
$\frac{mEI\_I}{mEI\_E}$ size	$\frac{32}{9082}$	0
EtoE time (sec)	40723.16	40255.53
EtoE time/elem/step (μsec)	18.68	18.47
Mflops/PE	254	284.2
L1 cache BW (Mb/sec)	1911.64	2362.14
L2 cache BW (Mb/sec)	500.29	511.38

The next three runs, presented in Table 5.7, compare mixed integration cases: one is the same case as the run in the previous batch with 1 PE executing mixed integration on one T2M3 system and 1023 PEs being purely explicit, second is with 5 PEs executing mixed integration of a single T2M3 and is foundation and 1019 PEs being explicit only and third is a case were one PE executes a large mixed integration and 1023 PEs being purely explicit. The first column of Table 5.7 when contrasted with the first column of Table 5.6 is the same type of run only using a different software stack. The performance is slightly better and this can be attributed to the MKL optimizations, the MVAPICH2 1.2 improved performance over MVAPICH 0.9.8 and the newer Intel compiler version. Also part of this is the fact that the simulations were conducted at different times on the supercomputer where the overall network traffic and task placement was different. The latter two effects can only be accounted for statistically by collecting a large number of data points regarding the performance of each run type. Regardless, the point is that the mixed integration performance has minimal software and communication (when within the same address space) overhead and is only as slow as the cost of the LU imposed on one or more PEs performing it. The time per element per time step per PE increase for the five versus the one PEs executing mixed integration of the same size is only 0.95% which shows that the mixed integration implementation scales well for the case of it being

limited within one PE per implicit domain and all of the implicit partitions being similar. The third case presented is one where the implicit partition on the PE is 10 times larger than the previous case, therefore a 1000 times slower in execution and hence the significant increase in the time per element per step per PE.

Table 5.7. T2M3 simulation performance results using MPVAPICH2 1.2, Intel 10.1.017, and MKL 10.0.3.020

# mixed PEs	1	5	1
# explicit PEs	1023	1019	1023
$\frac{mEI\_I}{mEI\_E}$ size	$\frac{32}{9082}$	$\frac{32}{9082}$	$\frac{320}{9082}$
EtoE time (sec)	38655.23	39024.61	67629.84
EtoE time/elem/step (μsec)	17.73	17.9	31.03
Mflops/PE	287.9	293.4	207.1
L1 cache BW (Mb/sec)	1913.57	2106.55	2748.19
L2 cache BW (Mb/sec)	537.65	528.33	326.13

In conclusion, the scalability is satisfactory with minimal performance overhead, and thus it allows for increasing the problem size and the number of PEs for a mixed integration problem.

Chapter 6

Conclusions and future directions

This final chapter summarizes the results and states the conclusions of the research. The conclusions are presented for both the computational and algorithmic and software design as well as on the engineering modeling and analysis of soil-foundation-structure interaction. Finally, future directions for research are outlined.

6.1 Summary and conclusions

The goal of this research was to develop a methodology and scalable software to simulate soil-foundation-structure interaction on a regional scale and investigate the local site effects, and the effect of near fault ground motion. To achieve these scientific goals the need to develop a high performance and scalability distributed memory software was identified and addressed.

The first step was to provide an improved seismic input over the traditional convolution to bedrock method and subsequently vertically propagating seismic waves. Software was developed for incorporation of the Domain Reduction Method in a finite element simulation. The DRM allows for three-dimensional input, consistent with boundary conditions, that allows for a subdomain to be analyzed as a body in equilibrium extracted

from a larger domain. The effects of the fault rupture, wave propagation path and basin effects are transfered and accounted for via the input wavefields.

To provide high fidelity simulation for soil domains using the domain reduction method a very scalable explicit linear finite element application was developed by developing loading pattern, system of equations, solver and degree-of-freedom numbering classes and also using components of the OpenSees framework. The software design and data structures allow for very high efficiency and to reach thousands of core counts in the MPI communicators while maintaining parallel efficiency of 95% for 4096 PEs. The communication model, used in a new solver class can be tuned depending on the problem size, hardware platform and MPI library. Several new classes were developed and existing ones from the framework were substantially modified in order to achieve the desirable performance levels. Emphasis was placed in both the numerical aspects and efficient memory management for the cache hierarchy.

Using the simulation tools, a site response analysis was performed for a 1 km by 0.5 km by 0.1 km region adjacent to the projection of the Puente Hills fault at a site in downtown Los Angeles, U.S.A. A scenario of a simulated M=7.1 earthquake was examined, which contained a large pulse shaking with low frequency ($\leq$ 0.5 Hz) spectral content. The input was generated by a collaborating group at Carnegie Mellon University. The simulations were carried out for the original soil profile (simplified as layered) and for a modified soft soil profile. Increased spatial variability was observed in the soft soil profile. In addition a high frequency component was generated stochastically by the same group and was used to enhance the low frequency deterministic input to a broadband one. The soil model was calibrated for the damping to be realistic and the output in terms of its spectral contents. More intense shaking was observed in the high frequency regions of the spectra for the broadband motion.

To enable the analysis of structures and their foundations, embedded in the soil domain, a mixed explicit-implicit integration algorithm was developed and implemented

based on an existing one, including extensions to allow for nonlinear systems in the implicit partition. The software design allows for use of the explicit components previously developed and thus incorporate, to a very high degree, their scalability and performance characteristics. The framework's architecture had to be bypassed or enhanced in order to allow for a more efficient implementation. A main conclusion is that the most computationally efficient approach was to implement the necessary synchronization/communication/exchange between the explicit and implicit partitions at the level of the systems of equations. This resulted in an economical implementation that avoided overheads from additional function calls and the creation of extra classes. The approach used is directly implementable in a non object oriented based code, such as a procedural based (FORTRAN) code.

Finally in the last chapter the tools and methodologies developed are used to analyze the same soil region enhanced with simplified models for the superstructure and foundation. Two types of simulation were performed: individual systems with two or four second vibration periods and varying yielding characteristics were examined, and also two regional simulations with five and six buildings were conducted to examine the spatial variability and building to building interactions. As far as the individual systems the results showed that the simplified solutions for SFSI tend to overestimate slightly the period lengthening as well as the damping reduction due to rocking. In most cases the structural deformations were reduced compared to the fixed base analyses and in few cases the permanent offsets were increased. The permanent deformation characteristics showed large variability for systems with the same fundamental period of vibration and varying yield strength. For the regional simulations with multiple buildings the two cases focused: one on the spatial variability of the response with buildings placed at 100 m apart and the other analysis examined the effects of the building to building interaction. For the input wavefield used and the structural properties selected the findings were that the differences between the individual system's response and the response due to multiple buildings were small. An overview of the scalability and performance characteristics of the analysis of

domains using mixed integration, utilizing data collected from the above analyses, shows that the design successfully maintained, to large extent, the performance characteristics of the pure explicit solution.

6.2 Future directions

This work has achieved the main objectives which were to develop efficient software that will enable the computational study of soil-foundation-structure interaction and to initiate the study of an entire urban region including near fault wave field effects as well as local site effects. Several future extensions and research directions related to both the software component as well as the engineering analysis have been identified.

The first point is related to software optimizations and tuning. While the algorithms designed and used have been carefully implemented, and standard programming techniques have been applied to render them efficient, there exists a plethora of optimizations that can be still applied such as: array padding, multilevel blocking, prefetching (software and hardware), SIMDization, and use of cache bypass by instruction which can potentially increase the arithmetic intensity and many more, other than just relying on the sophistication of the compilers. Some of these optimizations help reduce the memory traffic. Some of these are included as part of the SSE2 instruction set and on. More details can be found in Datta et al. (2008); Williams et al. (2008). It is expected that working towards this direction will improve the performance of the software. It is work that can easily be done on one processor, optimizing the serial performance, and the gains will be immediately transferable to the large scale runs.

The next point has to do with the size of the models considered in the simulations. While the solvers and performance will not be affected by increasing the total number of elements and PEs used, the need to produce very large meshes will render modifications to the meshing class for the subdomains necessary. More specifically, the interface to

the graph partitioning software will need to be augmented by interfacing to ParMetis in a distributed way, i.e. providing more than one submesh, the union of which will be partitioned in N subdomains. There exist other graph partitioning tools in the literature that have not been investigated as alternatives in this study such as SCOTCH (2007); Chaco (2007). The efficacy of these different graph partitioners has not been assessed as far as establishing whether they yield better balanced communication graphs with smaller number of crossing edges. Also a detailed study for selection of more advanced existing hexahedral meshing tools will be required instead of using the methods of the **Mesh3DSubdomain** class.

Studying linear soils begins to show the effect of site response but it does not reveal the entire range of possibilities. The next step is to extend the simulation methodology and software to nonlinear soil constitutive models. While the solution strategy can remain explicit for the soil, it is worthwhile examining an iterative implicit strategy for the entire domain. The conjugate gradient method is the obvious candidate for this problem and the effort should be focused on the preconditioning aspects of it.

The next research direction is to perform representative studies using wide ranges of parameters for the structural and foundation characteristics and focus on understanding the building to building interaction and when it is necessary to be accounted for. Most simulations to date are two-dimensional and the energy radiation in a three-dimensional analysis is different and needs to be understood together with its effect on the surface motion and structural responses.

Bibliography

Aagard, B. T., J. F. Hall, and T. Heaton (2001), Characterization of near-source ground motions with earthquake simulations, *Earthquake Spectra*, 17, 177–207, article.

Bao, H., J. Bielak, O. Ghattas, L. F. Kallivokas, D. R. O'Hallaron, J. Schewchuk, and J. Xu (2001), Large-scale simulation of elastic wave propagation in heterogeneous media on parallel computers, *Comput. Methods Appl. Mech. Eng.*, 152, 85–102, article.

Bathe, K.-J. (1996), Finite Element Procedures, Prentice Hall, Upper Saddle River, New Jersey, U.S.A.

Belytschko, T., W. K. Liu, and B. Moran (2000), Nonlinear Finite Elements for Continua and Structures, Wiley.

Bielak, J. (1975), Dynamic behavior of structures with embedded foundations., *J. Earthquake Engrg. Struct. Dyn.*, 3(3), 259–274, article.

Bielak, J., K. Loukakis, Y. Hisada, and C. Yoshimura (2003a), Domain reduction method for three-dimensional earthquake modeling in localized regions. part i: Theory, *Bull. Seism. Soc. Am.*, 93(2), 817–824, article.

Bielak, J., K. Loukakis, Y. Hisada, and C. Yoshimura (2003b), Domain reduction method for three-dimensional earthquake modeling in localized regions. part ii: Verification and applications, *Bull. Seism. Soc. Am.*, 93(2), 825–840, article.

Chaco (2007), Chaco: Software for partitioning graphs, http://www.cs.sandia.gov/ bahendr/chaco.html.

Chopra, A. K. (2001), Dynamics of Structures: Theory and Applications to Earthquake Engineering, Prentice Hall, Englewood Cliffs, New Jersey, U.S.A.

Chopra, A. K., and C. Chintanapakdee (2001), Comparing response of sdf systems to near-fault and far-fault earthquake motions in the context of spectral regions, *Earthquake Engineering and Structural Dynamics*, 30, 1769–1789.

Cook, R. D., D. S. Malkus, and M. E. Plesha (1988), Concepts and Applications of Finite Element Analysis, John Wiley & Sons.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein (2001), Introduction to algorithms, MIT Press, Cambridge, MA, USA.

Courant, R., K. Friedrichs, and H. Lewy (1967), On the partial difference equations of mathematical physics, *IBM Journal*, pp. 215–234.

Danielson, K. T., and R. R. Namburu (1998), Nonlinear dynamic finite element analysis on parallel computers using FORTRAN 90 and MPI, *Advances in Engineering Software*, 29(3-6), 179–186.

Datta, K., M. Murphy, V. Volkov, S. Williams, J. Carter, L. Oliker, D. Patterson, J. Shalf, and K. Yelick (2008), Stencil Computation Optimization and Auto-tuning on State-of-the-Art Multicore Architectures, *Proceedings of Supercomputing (SC)*.

Davis, P. M., J. L. Rubenstein, K. H. Liu, S. S. Gao, and L. Knopoff (2000), Northridge earthquake damage caused by geologic focusing of seismic waves, *Science*, 289, 1746–1750, article.

Demmel, J. W. (1997), Applied Numerical Linear Algebra, SIAM.

Elgamal, A., L. Yan, Y. Zhang, and J. P. Conte (2008), Three-dimensional seismic response analysis of humbolt bay bridge-foundation-ground system, *Journal of Structural Engineering*, 134(7), 1165–1176, article.

FEMA (2008), FEMA's Software Program for Estimating Potential Losses from Disasters, http://www.fema.gov/plan/prevent/hazus.

Fenves, G. L. (1990), Object-oriented programming for engineering software development, *Engineering With Computers*, 6(1), 1–15.

Forum, M. P. I. (1994), MPI: A Message-Passing Interface Standard, http://www.mpi-forum.org/.

Gamma, E., R. Helm, R. Johnson, and J. Vlissides (1995), Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional Computing Series.

Golub, G. H., and C. F. Van Loan (1996), Matrix Computations, Johns Hopkins.

Graves, R. W. (1993), Modeling three-dimensional site response effects in the marina district basin, san franncisco, california, *Bull. Seism. Soc. Am.*, 83, 1042–1063, article.

Graves, R. W., A. Pitarka, and P. G. Somerville (1998), Ground motion amplification in the santa monica area: effects of shallow basin edge structure, *Bull. Seism. Soc. Am.*, 88, 1224–1242, article.

Gravouil, A., and A. Combescure (2001), Multi-time-step explicit-implicit method for non-linear structural dynamics, *International Journal for Numerical Methods in Engineering*, 50, 199–225, article.

Hairer, E., S. Nørsett, and G. Wanner (1993), Solving Ordinary Differential Equations I. Nonstiff Problems, Springer-Verlag.

Hairer, E., S. Nørsett, and G. Wanner (2004), Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems, Springer-Verlag.

Hughes, T. J. R. (2000), Linear Static and Dynamic Finite Element Analysis, Dover Publications, Mineola NY.

Hughes, T. J. R., and W. K. Liu (1978a), Implicit-explicit finite elements in transient analysis: Stability theory, *Journal of Applied Mechanics*, 45, 371–374, article.

Hughes, T. J. R., and W. K. Liu (1978b), Implicit-explicit finite elements in transient analysis: Implementation and numerical examples, *Journal of Applied Mechanics*, 45, 375–378, article.

Hughes, T. J. R., K. S. Pister, and R. L. Taylor (1979), Implicit-explicit finite elements in nonlinear transient analysis, *Computer Methods in Applied Mechanics and Engineering*, 17(18), 159–182, article.

IBM (2006), Engineering Scientific Subroutine Library, http://www-03.ibm.com/systems/p/software/essl.html.

Ichimura, T., and M. Hori (2000), Macro-micro analysis for prediction of strong motion distribution in metropolis, *J. Struct. Eng./Earthquake Eng., JSCE*, I-52(654), 51–62, article.

Idriss, I. M., and J. I. Sun (1992), SHAKE91: A computer program for conducting equivalent linear seismic response analyses of horizontally layered soil deposits, Center for Geotechnical Modeling, U.C. Davis.

INTEL (2008), Intel Math Kernel Library, http://www.intel.com/cd/software/products/asmo-na/eng/307757.htm.

Karypis, G. (1998), METIS - serial graph partitioning and fill-reducing matrix ordering,.

Liu, W. K., and T. Belytschko (1982), Mixed-time implicit-explicit finite elements for transient analysis, *Computers and Structures*, 15(4), 445–450, article.

Luco, J. E. (1980), Linear soil-structure interaction. soil-structure interaction: The status of current analysis methods and research, *Rep. No. NUREG/CR-1780 and UCRL-53011, U.S. Nuclear Regulatory Commission, Washington, D.C. and Lawrence Livermore Laboratory, Livermore, Calif.*, report.

Lysmer, J., and R. L. Kuhlemeyer (1969), Finite dynamic model for infinite media., *Journal of the Engineering Mechanics Division, ASCE*, 95(3), 859–877, article.

McKenna, F. T. (1997), Object oriented finite element programming: Framework for analysis, algorithm and parallel computing, Ph.D. thesis, University of California, Berkeley.

McKenna, F. T., and G. L. Fenves (2004), Opensees: Open system for earthquake engineering simulation, http://opensees.berkeley.edu.

MVAPICH: (2008), MPI over Infiniband and iWARP http://mvapich.cse.ohio-state.edu/.

NCSA (2008), National Center for Supercomputing Applications at the University of Illinois, http://www.ncsa.uiuc.edu/index.html.

Neuenhofer, A., and F. C. Filippou (1998), Geometrically nonlinear flexibility-based frame finite element, *ASCE Journal of Structural Engineering*, 124(6), 704–711.

Newmark, N. M. (1959), A method of computation for structural dynamics, *Journal of the Engineering Mechanics Division*, 85, 67–94, article.

Olsen, A. H., B. T. Aagaard, and T. H. Heaton (2008), Long-period building response to earthquakes in the san francisco bay area, *Bull. Seism. Soc. of Am.*, 98(2), 1047–1065.

Olsen, K. B. (2001), Three-dimensional ground motion simulations for large earthquakes on the san andreas fault with dynamic and observational constraints., *Journal Comp. Acoust.*, 9(3), 1203–1215, article.

Olsen, K. B., S. M. Day, and C. R. Bradley (2003), Estimation of q for long-period (>2 sec) waves in the los angeles basin, *Bull. Seism. Soc. Am.*, 93(2), 627–638, article.

PAPI (2008), Performance Application Programming Interface http://icl.cs.utk.edu/papi.

Park, J. S. (2004), Parallel simulation of structural performance in earthquakes, Ph.D. thesis, University of California, Berkeley.

Park, J. S., G. L. Fenves, and B. Stojadinovic (2004), Spatial distribution of response of multi-story structures for simulated ground motions, *Proceedings of 13th World Conference on Earthquake Engineering*, pp. 13wcee–001,545, article.

PerfSuite (2008), http://perfsuite.ncsa.uiuc.edu.

Richtmyer, R. D., and K. W. Morton (1957), Difference methods for initial-value problems, krieger Publishing company, Krieger Drive, Malabar Florida.

Roeset, J. M. (1980), A review of soil-structure interaction. soil-structure interaction: The status of current analysis methods and research, *Rep. No. NUREG/CR-1780 and UCRL-53011, U.S. Nuclear Regulatory Commission, Washington, D.C. and Lawrence Livermore Laboratory, Livermore, Calif.*, report.

SCOTCH (2007), Software package and libraries for graph, mesh and hypergraph partitioning, static mapping, and parallel and sequential sparse matrix block ordering http://www.labri.fr/perso/pelegrin/scotch/.

SDSC (2006), San Diego SuperComputing Center at U.C.S.D., http://www.sdsc.edu/ .

Semblat, J. F., M. Kham, P.-Y. Bard, and P. Gueguen (2004), Could "site-city interaction" modify site effects in urban areas?, *13th World Conference on Earthquake Engineering*, Conference Proceedings(1978), paper.

Stewart, J. P., G. L. Fenves, and R. B. Seed (1999a), Seismic soil-structure interaction in buildings. i: Analytical methods, *Journal of Geotechnical and Geoenvironmental Engineering*, 125(1), 26–37, article.

Stewart, J. P., G. L. Fenves, and R. B. Seed (1999b), Seismic soil-structure interaction in buildings. ii: Experimental methods, *Journal of Geotechnical and Geoenvironmental Engineering*, 125(1), 38–48, article.

Stewart, J. P., J. D. Bray, S.-J. Chiou, R. W. Graves, P. G. Somerville, and N. A. Abrahamson (2001), Ground motion evaluation procedures for performance-based design, Tech. rep., Pacific Earthquake Engineering Research Center.

Toselli, A., and O. Widlund (2005), Domain Decomposition Methods - Algorithms and Theory, Springer.

Veletsos, A. S., and J. W. Meek (1974), Dynamic behavior of building-foundation systems, *J. Earthquake Engrg. Struct. Dyn.*, 3(2), 121–138, article.

Veletsos, A. S., and V. V. Nair (1975), Seismic interaction of structures on hysteretic foundations, *J. Struct. Engrg.*, 101(1), 109–129, article.

Williams, S., J. Carter, L. Oliker, J. Shalf, and K. Yelick (2008), Lattice Boltzmann Simulation Optimization on Leading Multicore Platforms, *International Parallel & Dstributed Processing Symposium (IPDPS).*

Zhang, Y., J. P. Conte, A. Elgamal, Bielak, and G. Acero (2008), Two-dimensional nonlinear earthquake response analysis of a bridge- foundation-ground system, *Earthquake Spectra*, 24(2), 343–386, article.

Appendix A

C++ Header Files

A.1 Public Members of Mesh3DSubdomain Class

Listing A.1. Mesh3DSubdomain Class

```
class Mesh3DSubdomain {
  public :
  //general constructor
  Mesh3DSubdomain(Domain * inpDomain) ;
  Mesh3DSubdomain() ;
  //destructor
  virtual ~Mesh3DSubdomain();
  void make3DUniformMesh(int firstNewNodeNumber, int firstNewEleNum,
                         double startX, double startY, double startZ,
                         int nx, int ny, int nz,
                         double Lx, double Ly, double Lz,
                         int* elmnts, int* ew, double* nds, int* nw);
  void write3DUniformMeshToIFile(int* elmnts, int size_elmnts,
                                 int* epart, int metis_ne,
                                 double* nds, int size_nds,
                                 int metis_nn, int who_am_I,
                                 double* Vs1, double* nu_soil,
                                 double* soil_density,
                                 double* alphaM, double* betaK,
                                 double* betaK0, double* betaKc,
                                 int numLayers, double* layers,
                                 int* ne, int* nn, int* nl,
                                 double xMin, double xMax,
                                 double yMin, double yMax,
                                 double zMin, double zMax,
                                 double Lx, double Ly, double Lz);
  void write3DUniformMeshForAll(int* elmnts, int size_elmnts,
                                int* epart, int metis_ne,
                                double* nds, int size_nds,
                                int metis_nn, int numProcesses,
```

```
                                    double* Vs1, double* nu_soil,
                                    double* soil_density,
                                    double* alphaM, double* betaK,
                                    double* betaK0, double* betaKc,
                                    int numLayers, double* layers,
                                    double xMin, double xMax,
                                    double yMin, double yMax,
                                    double zMin, double zMax,
                                    double Lx, double Ly, double Lz);
  void read3DUniformMeshFromIFile(int who_am_I,
                                  int numProcs, bool makeLysmers);
  void allocate_e_Nodes(double xMin, double xMax,
                        double yMin, double yMax,
                        double zMin, double zMax,
                        std::map<int,int>& eNodes);
  void allocateBoundaryLayerElements(double xMin, double xMax,
                                     double yMin, double yMax,
                                     double zMin, double zMax,
                                     std::map<int,Element*>& elements);
};
```

A.2 Public Members of PlaneDRMInputHandler Class

Listing A.2. PlaneDMRInputHandler Class

```
class PlaneDRMInputHandler : public DRMInputHandler {

 public:

  PlaneDRMInputHandler(int tag, char** in_files, int files,
                       double dt, double* time_array, int num_steps,
                       int* file_data, int fileData_size,
                       double* domain_crds, double* drm_box_crds,
                       double* eleD, Mesh3DSubdomain* my_mesher,
                       int steps_to_cache, Domain* domain);

  virtual ~PlaneDRMInputHandler();

  void populateBuffers();
  void getMotions(Element* eletag, double time,
                  Vector& U, Vector& Ud, Vector& Udd);
  void computeHistory(Element* eletag, double time,
                      Vector& U, Vector& Ud, Vector& Udd,
                      bool updateDm1);
  void handle_elementAtface5(Element* eletag, double time,
                             Vector& U, Vector& Ud, Vector& Udd);
  void handle_elementAtface1(Element* eletag, double time,
                             Vector& U, Vector& Ud, Vector& Udd);
  void handle_elementAtface2(Element* eletag, double time,
                             Vector& U, Vector& Ud, Vector& Udd);
  void handle_elementAtface3(Element* eletag, double time,
                             Vector& U, Vector& Ud, Vector& Udd);
  void handle_elementAtface4(Element* eletag, double time,
                             Vector& U, Vector& Ud, Vector& Udd);
  void getLocations(double x, double y,
                    double dx, double dy, int* xloc, int* yloc);
  void getTemporal(double time, int* tloc);
  int getIndex(double time);
  void getf5pointer(Node* node_tag, int local_tag, int index);
  void getf1pointer(Node* node_tag, int local_tag, int index);
  void getf2pointer(Node* node_tag, int local_tag, int index);
  void getf3pointer(Node* node_tag, int local_tag, int index);
  void getf4pointer(Node* node_tag, int local_tag, int index);
  void pointerCopy(int node_from, int node_to);
  void populateTempBuffers(int index, int fileptr,
                           double ksi, double eta);
};
#endif
```

A.3 Public Members of GeometricBrickDecorator Class

Listing A.3. GeometricBrickDecorator Class

```
class GeometricBrickDecorator {
  public :
  GeometricBrickDecorator();
  virtual ~GeometricBrickDecorator();
  void setBrick(Element* elem);
  void setDomain(Domain * theDomain);
  void clearBrick();
  void clearDomain();
  void clearAll();
  bool isClear();
  bool isEmpty();
  bool isZero(double a, double b);
  bool isPointInVolume(const Vector &pP);
  double evalSignedVol(const Vector &pA,
                       const Vector &pB,
                       const Vector &pC,
                       const Vector &pP);
  double eval3OrderDet(const Vector &pA,
                       const Vector &pB,
                       const Vector &pC);
  void getFace(int which, ID& face, ID& faceID);
  double getMinMaxCrds(int which, int whichtoret);
  bool compareFaceToFace(int which, ID &faceOther);
  bool isFaceinPlane(int which, const Vector &pP);
  bool isFaceinVertPlane(int which, double xy,
                         double zmin, double zmax,
                         int whichCrd);
  bool isLeftBoundary(double xMin, double xMax,
                      double yMin, double yMax,
                      double zMin, double zMax);
  bool isRightBoundary(double xMin, double xMax,
                       double yMin, double yMax,
                       double zMin, double zMax);
  bool isFrontBoundary(double xMin, double xMax,
                       double yMin, double yMax,
                       double zMin, double zMax);
  bool isRearBoundary(double xMin, double xMax,
                      double yMin, double yMax,
                      double zMin, double zMax);
  bool isBottomBoundary(double xMin, double xMax,
                        double yMin, double yMax,
                        double zMin, double zMax);
  bool isBoundaryLayerEle(double xMin, double xMax,
                        double yMin, double yMax,
                        double zMin, double zMax);
}
```

A.4 Public Members of DRMBoundaryLayerDecorator Class

Listing A.4. DRMBoundaryLayerDecorator Class

```
class DRMBoundaryLayerDecorator {
  friend class Element;
 public :
  DRMBoundaryLayerDecorator();
  virtual ~DRMBoundaryLayerDecorator();
  void setMap(std::map<int,int> &eNodes);
  void setSet(std::set<int,std::less<int >> &eNodes);
  void setDomain(Domain *theDomain);
  void clearDomain();
  void setBrick(Element* brickTag);
  void clearBrick();
  void clear();
  void computeDRMLoad(Vector &drmLoad,
                      const Vector &displ,
                      const Vector &veloc,
                      const Vector &accel);
  void applyDRMLoad(Vector &drmLoad,
                    const Vector &displ,
                    const Vector &veloc,
                    const Vector &accel);
  void get_E_B_Nodes(ID &e, ID &b, std::map<int,int> &eNodes);
};
```