Earthquake-Soil-Structure Interaction Modeling of Nuclear Power Plants for Near-Field Events

By

José Antonio Abell Mena
B.S. (Pontificia Universidad Católica de Chile) 2007
M.S. (Pontificia Universidad Católica de Chile) 2008

Dissertation

Submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Civil & Environmental Engineering

in the

Office of Graduate Studies

of the

University of California

Davis

Approved:

Professor Boris Jeremić, Chair

Professor Sashi K. Kunnath

Professor Zhaojun Bai

Committee in Charge

2016

Contents

Contents ii

List of Figures iv

List of Tables ix

Abstract x

Acknowledgments xi

Chapter 1. Introduction 1

1.1. Hypothesis 4

1.2. Scope of Study 4

1.3. Original Features 5

1.4. Summary of Contents 6

Chapter 2. Methods for Earthquake-Soil-Structure Interaction Analysis 8

2.1. Soil-Structure Interaction Modeling: State-of-Practice 8

2.2. The Domain Reduction Method 12

2.3. Review of Previous Research on Using DRM for ESSI 16

Chapter 3. Earthquake-Soil-Structure-Interaction Simulation Platform 18

3.1. Classical Elasto-Plasticity Implemented Using C++11 Template Metaprogramming 18

3.2. Domain Specific Language for Finite Element Modeling and Simulation 47

3.3. Parallel Finite Element Method 59

Chapter 4. Generation of Seismic Motions by Physical Modeling 65

4.1. Problem statement 65

4.2. Conditions for correct solution of the elastodynamic wave equation 69

4.3. Earthquake source characterization 72
4.4. Postulating new earthquake scenarios 78
4.5. A study on wave propagation in layered a medium 81
4.6. Development of Seismic Wave Fields for ESSI Modeling 89

Chapter 5. Response of a Soil-Structure System to 3–D and 1–D Representations of the Seismic Wave-Field 99
5.1. Problem Statement 99
5.2. Finite-element mesh of the NPP and neighboring soil 100
5.3. Non-linear soil model 102
5.4. Earthquake source time-function 104
5.5. Generated surface motion and 1–D Site deconvolution 107
5.6. Results: Deep Source. Linear Model 110
5.7. Results for Shallow Source Input. Linear vs Nonlinear Models 111
5.8. Discussion of Results for Shallow Source, Linear vs. Non-Linear Models 115
5.9. Results for Shallow Source Input. 3–D vs 3× 1–D Seismic Wave Fields 117
5.10. Discussion of Results for Shallow Source, 3–D vs. 3×1–D Seismic Wave Fields 121
5.11. Energy Dissipation Distribution Results 122
5.12. Discussion of Energy Dissipation Distribution Results 126

Chapter 6. Summary 127
6.1. Future Lines of Research 128

Appendix A. Review of the Non-Linear Finite-Element Method for Small-Deformations 130
A.1. Finite-element method for small deformation dynamic problems with material non-linearity 130

Appendix B. Example of Analysis Using the FEI 139
B.1. Triaxial test 139
B.2. 1–D Wave propagation 142

Bibliography 149

List of Figures

2.1 Complete ESSI problem consists of source and path modeling as well as local site and structural response. 12

2.2 Complete ESSI problem consists of source and path modeling as well as local site and structural response. 14

3.1 A UML class diagram for the relationships between the main classes in the framework. 35

3.2 Configuration of the single element test for verification of the elasto-plasticity models developed within the framework. 42

3.3 The stress pseudo-time history applied to the element after confinement to $p_0 = -250\,[\text{kPa}]$ and the resulting shear-strain pseudo-time history. 44

3.4 Stress-controlled response of the material models to a sawtooth (triangular) shaped shear-stress loading after confinement with $\sigma_{zz} = -250\,[\text{kPa}]$ 45

3.5 Implied modulus reduction curve G/G_{max} for the resulting material response. 46

3.6 Connectivity criterion for 8 node bricks. 60

3.7 Finite element mesh and its dual graph 60

3.8 Visualization of a typical `METIS` partition of a 2-D mesh into 25 partitions. 62

3.9 Typical stiffness matrix for a FEM mesh, showing how distributing rows amongst processors generates communication between them. 63

4.1 Linear interpolation of a sine wave with increasing number of points (N_{pts}). Note how $N_{pts} = 7$ completely misses the peaks. 71

4.2 Sine-wave interpolation error for varying number of points and integration schemes. Doubling interpolation order can yield up to two orders of magnitude error decrease. Alternatively, doubling integration order allows the use of sample spacing twice as large. 72
4.3 Fault geometry for a simple planar fault. 75
4.4 An extended fault discretized in many planar faults 77
4.5 A fault is discretized into various point sources. The contributions of each subfault are summed at the site of interest. 78
4.6 (Left) Not to scale, basic geometry of the area modeled. (Right) Coordinate system and location of earthquake source and receiver stations. 83
4.7 Material distributions (a) layered, (b) layered with intrusion on left of the domain, and (c) layered with intrusion to the right of the domain. 83
4.8 Gaussian pulse used in this work. 84
4.9 X-component of response in depth at the site. 84
4.10 Z-component of response in depth at the site. 85
4.11 Traces of particle motions around the site of interest for different source depths for the layered geology. Line-color shows time and layering is also shown for reference. Note that only the deep source can be considered to produce approximately vertically propagating S-Wave. 86
4.12 Particle motion traces for the case of intrusion located at the right of the domain. 87
4.13 Particle motion traces for the case of intrusion located at the left of the domain. 88
4.14 Model layout and coordinate system origin and orientation for (a) SW4 (b) ESSI Simulator 90
4.15 Free field DRM model for verification of SW4 and ESSI Simulator coupling. From top, internal domain, DRM boundary, absorbent element layer, complete model. 95
4.16 Visualization of the DRM solution at time $t = 3.40\,[\mathrm{s}]$ for DRM mesh sizes of $h = 20\,[\mathrm{m}]$ (left) $10\,[\mathrm{m}]$ (right), SW4 mesh is $h = 20\,[\mathrm{m}]$ for both cases. Color shows magnitude of displacement vector. Arrival of P wave is correctly resolved on both meshses 96

4.17 Visualization of the DRM solution at time $t = 3.93\,[\mathrm{s}]$ for same setup as Figure 4.16, with color showing magnitude of the displacement vector. Arrival of S wave is better resolved on the $h = 10\,[\mathrm{m}]$ mesh but not on the $h = 20\,[\mathrm{m}]$ mesh as can be seen by looking at the out-going motions. SW4 mesh is $h = 20\,[\mathrm{m}]$ for both cases 96

4.18 Generated motions at the center node ($x = 4000m$, $y = 2000$, $z = 0$), blue shows SW4 motions generated with an $h = 2\,[\mathrm{m}]$ grid, and orange shows motions obtained with RealESSI using DRM and variable mesh size. 97

4.19 Results from computing motions with ESSI Simulator within the DRM domain for different ESSI mesh sizes h and Rayleigh damping with $\xi = 0.1$. All SW4 motions were computed for $h = 20\,[\mathrm{m}]$ 98

5.1 Nuclear power plant geometry 100

5.2 FEM mesh of the site. 101

5.3 FEM mesh of the site and NPP featuring, in orange, the layer of DRM elements. 103

5.4 Response of material to symmetric and unsymmetric loading showcasing the ratcheting due to the Bauschinger effect 103

5.5 Stiffness degradation response of the material 104

5.6 Corner frequency vs. moment magnitude for different values of the stress drop. 105

5.7 Time and frequency plots for a Brune and BruneSmoothed source time functions both for $f_0 = \frac{\omega_0}{2\pi} = 6.2\,[\mathrm{Hz}]$ 106

5.8 Displacement and acceleration histories for the generated earthquake motions at the NPP site. 108

5.9 Fourier amplitude spectrum and 5%-damped acceleration response spectrum for the generated earthquake motions at the NPP site. 109

5.10 Displacement response histories at the top of foundation and containment building for seismic motions coming from a deeps source. Linear results. 110

5.11 Structural response plots at the top of foundation comparing linear response to non-linear response, when subjected to 3–D motions. 111

5.12 Structural response plots at the top of containment comparing linear response to non-linear one, when subjected to 3–D motions. 112

5.13 Structural response plots at the top south-west corner of auxiliary building comparing linear response to non-linear response, when subjected to 3–D motions. 113

5.14 Structural response plots at the top north-west corner of auxiliary building comparing linear response to non-linear response, when subjected to 3–D motions. 114

5.15 Deformed shape of the NPP and isolated foundation slab at peak displacement for the non-linear 3–D input case. Colorbar displacements in meters. 116

5.16 Structural response of the Non-linear sPP sub- jected to 3–D motions versucase when de-convolved 1–D, computed motions are used plots . The same non-linear model for soil was used. 117

5.17 Non-linear structural response of the NPP subjected to 3–D motions versus the case when 1–D motions are used plots, computed at the top of containment comparing linear response 118

5.18 Non-linear structural response of the NPP subjected to 3–D motions versus the case when 1–D motions are used plots, computed at the top of south-west corner of auxiliary comparin. 119

5.19 Non-linear structural response of the NPP subjected to 3–D motions versus the case when 1–D motions are used plots, computed at the top of north-west corner of auxiliary building. 120

5.20 Volume render visualization of plastic energy dissipation rate in $\mathrm{kJ/s/m^3}$ at t = 0.86 sec for soil under the NPP. **(Top)** result for full 3–D simulation. **(Bottom)** result for full 3×1–D simulation. 122

5.21 Volume render visualization of plastic energy dissipation rate in $\mathrm{kJ/s/m^3}$ at t = 1.57 sec for soil under the NPP. **(Top)** result for full 3–D simulation. **(Bottom)** result for full 3×1–D simulation. 123

5.22 Volume render visualization of plastic energy dissipation rate in $\mathrm{kJ/s/m^3}$ at t = 1.78 sec for soil under the NPP. **(Top)** result for full 3–D simulation. **(Bottom)** result for full 3×1–D simulation. 124

5.23 Volume render visualization of plastic energy dissipation rate in $kJ/s/m^3$ at t = 2.14 sec for soil under the NPP. **(Top)** result for full 3–D simulation. **(Bottom)** result for full 3×1–D simulation. 125

A.1 Illustration of the domain within which the dynamic elasto-plastic equations of motion are to be solved. 131

B.1 Single element triaxial test model setup. 139
B.2 Example 1, Triaxial test. Definition of model parameters. 140
B.3 Example 1, Triaxial test. Adding the material to the domain. 140
B.4 Example 1, Triaxial test. Definition of eight nodes and the brick element. 141
B.5 Example 1, Triaxial test. Boundary conditions 141
B.6 Example 1, Triaxial test. Confinement stage. 142
B.7 Example 1, Triaxial test. Simulation options. 142
B.8 Example 1, Triaxial test. Second stage. 143
B.9 Example 2, 1–D Wave propagation simulation. This listing set the global parameters that control the simulation. 144
B.10 Example 2, 1–D Wave propagation simulation. Automatic generation of material properties, nodes, elements, and constrains based on global parameters. 145
B.11 Example 2, 1–D Wave propagation simulation. Input motion is set as a 'imposed motion' boundary condition and the simulation is setup and executed 146

List of Tables

3.1 Definitions of `double function_to_compute(const DTensor2 & sigma)` for each case. 31

3.2 Relative speedup of using Static Polymorphism enabled through CRTP vs. Dynamic Polymorphism for different compiler optimization levels. 34

3.3 A non-exhaustive list of some FEA programs and their modeling paradigms. 48

3.4 Logical operations defined in ⊞ 53

4.1 Mesh information for DRM models used with RealESSI simulator. 91

B.1 List of predefined units in ⊞ 147

B.2 Second-order keywords defined in ⊞ 148

Abstract

This dissertation proposes an approach to modeling the response of a nuclear power facility considering soil-structure interaction, when subjected to earthquake motions originated in the near-field. It is argued that near-field earthquake-induced motions are complex in the sense that current state-of-practice assumptions made on the nature of seismic wave-field stemming from such events are oversimplified. Furthermore, even if near-field sources might not deliver the largest magnitude earthquakes for a given seismic setting, it is possible that the intensity of motions generated by such sources controls design of structural and/or non-structural components of nuclear facilities in some frequency range. Several nuclear power facilities are located in the vicinity of known smaller earthquake sources (within less than 10 [km]).

The domain reduction method is used to excite a model of the soil-structure system with a three-dimensional seismic wave-field which is computed using a state-of-the-art seismic simulation code. The response of this model is compared with that of an alternative model which assumes that the incoming wave-field is not three-dimensional but unidimensional. This last modeling approach is the most common in both the research and practice of nuclear power-plant seismic design. Two source-to-site geometries are evaluated to compare possible effects of the propagation path.

Computation of non-linear soil response is achieved by using a new implementation of the classical elasto-plasticity constitutive modeling framework using the new language features of the C++11 standard. This novel implementation scheme aims at being both efficient and maintainable by software-engineering standards. Both these goals are hard to achieve with just the features of previous editions of the C++ standard.

Acknowledgments

I'd like to express my deepest gratitude to my advisor Professor Boris Jeremić. His insight, guidance, support, passion for excellence and cheerfulness have made me grow in ways I never thought I could. Most of all I'd like to thank him for his tremendous understanding and caring for the complexities I've found myself in by being a Ph.D. student, a husband and a father. Thank you!

Much gratefulness is due to my dissertation committee in charge, Professor Dr. Sashi Kunnath and Professor Dr. Zhaojun Bai. Thank you for reviewing and appreciating my work, as well as for your feedback.

To my sponsors, Universidad de los Andes in Chile and also the Chilean government doctoral scholarship program 'Becas Chile'. It is my hope that your support will be retributed by my work back in my country, and that I will contribute to the growth and prosperity of my country.

For the friendship and technical help I received from students and scholars I met at UCDavis. Nima, Federico, Alexandros, Gregor, and others. Talking and working with you has contributed significantly to my work. Also to my brother-in-law Jorge, from whom I've gained much insight into seismology and much help in times of need.

To my parents, who have made very hard decisions and have struggled for me to pursue my dreams. Ultimately I am here in great measure because of you. Thank you.

To my parents-in-law, who have treated me like a son since the beginning. Thank you for your care, support and love.

Last, but most heartfeltedly, to my dear wife, Claudia. Thank you for walking next to me in this adventure. Thank you for you unconditional love. Thank you for for being the source of my joy and my true inspiration. This work is dedicated to you and to my children.

Written in Davis, California, March 18 of the year 2016.

CHAPTER 1

Introduction

In soil-structure interaction (SSI) modeling and analysis of civil infrastructure there are three main sources of modeling uncertainties: (i) geometric uncertainty, (ii) material response uncertainty, and (iii) earthquake loading uncertainty. Geometric uncertainty is the concept that the layout of materials as idealized mathematically will differ from the layout as realized upon construction of the infrastructure object. Material response uncertainty stems from choosing a material model which only captures–with limited accuracy–some of the observed behaviors and leaves out others for simplification and tractability. Additionally, due to the variability of behaviors observed in materials (especially soils) within a site of interest, the optimal numerical parameters describing such models are known at best within an interval of confidence. Lastly, earthquake loading uncertainty concerns the analyst with the nature of the incoming seismic wave field and its fundamental properties, such as duration, frequency content, spatial distribution, etc.; all of which affect the response of the studied object in ways that cannot be easily anticipated.

Both researchers and practitioners in SSI analysis subject to earthquake loading have payed a great deal of attention to issues of non-linear response of soil and structure, that is, material response uncertainty. Indeed, material response can be tested in the lab or inferred from field measurements. Constitutive modeling techniques can, then, capture the most relevant aspects of material response to the expected loading paths, given that sufficient testing is provided to calibrate these models.

Geometric uncertainty is usually dealt with by adjusting models when significant changes occur in the projected structure, or as new information on the site becomes available.

Earthquake uncertainty has many aspects to it. The focus has been to try to predict the level of earthquake-induced motions at a site through ground motion prediction equations (GMPEs,

which are based on statistics of recorded motions and, recently, on numerical simulations). GMPEs give an estimate and a range of the observed peak intensity, duration, and frequency content of the incoming seismic motion as a function of earthquake magnitude, distance, faulting style, and some site characteristics. Largely unaddressed, owing to the difficulty of the task, are the spatial and temporal variability of the earthquake motions within the site. Spatiotemporal variability is known be important for certain types of soil-structure systems, but their importance cannot be anticipated simply for an arbitrary case.

A major simplifying assumption that is commonly made to avoid dealing with the spatiotemporal nature of earthquake motions is to *assume* that the seismic wave-field propagates into the site as a vertically-propagating field composed of compressional and shear waves. This assumption is referred herein as the '1–D site' assumption, owing to the fact that such a field is only dependent on depth under the surface and time. The usual procedure to input these motions into a model of the site and structure has three stages. First, the analyst must select a set of seismic records (or synthetic motions) which are compatible with the scenario predicted by the GMPEs (usually processing the records to improve match). Second, a unidimensional site response analysis (e.g., without the structure) is performed such that an input motion can be determined at the base of the model which will produce the selected motion records at the site surface. This process is called 'ground motion record deconvolution'. Finally, the deconvolved motion is input into a complete model of site and structure and a full SSI analysis is performed. Several methods are available to the analyst as to how to input these motions depending on whether it is expected that structure-induced motions will radiate out of the domain, which is usually the case, leading to energy dissipation through radiation. If all three components of ground motion are thus processed this type of analysis is termed, herein, as '3×1–D' analysis.

There are many reasons why this assumption is rationally incorrect. For example, it implies that vertical motions at a site are produced exclusively by vertically propagating compressional waves, when it is known that these produce only a small fraction of the vertical component at the time of the first arrivals of waves into a site. Most of the vertical component is produced mainly by inclined shear waves and, later on, by surface waves. Furthermore, the 1–D site model predicts no surface waves whatsoever. It also predicts that the arrival of waves into the site is as

a wave where horizontal layers experience the same motions in phase until they are disturbed by reflections from structural interaction. The reality is that the motions within the soil arrive from all possible incidence angles, as well as as surface waves. This means that motions will not at all be in phase for horizontal planes (or any other plane orientation for that matter), and that this phenomenon will only increase with frequency. Coherence can only be argued for when the wavelengths of the incoming motions are comparatively large when compared to site size. This occurs at very low frequencies and/or at high propagation speeds. Even then, surface waves will still offset the wave field in depth .

It can't be argued, though, that the 1–D assumption has played a crucial role practical studies of SSI and has enabled the field to advance. It remains a simple means to use available data for SSI studies, which could otherwise never been done. It is also attractive in that it allows using real recorded motions to excite the model, although this is not as realistic as it is commonly assumed.

The difficulty in avoiding the 1–D site assumption lies in that it requires knowledge of the seismic wave-field everywhere within the site boundaries, even in depth. Instrumenting to obtain such information is prohibitively expensive as it requires devoting a large number of broadband seismometers to a single structure. These instruments could be put to better use as a network to measure earthquake motions at a regional scale, aimed at improving the amount of data available to study the earthquake phenomenon. Alternatively, numerical modeling can be used to obtain a detailed representation of the seismic wave-field within the site, so long as there is established confidence in the suitability of the seismic wave propagation modeling methodologies.

Even if trustworthy numerical modeling of the wave-field is available, the computational cost of performing an analysis of these characteristics is high. Digitally storing a detailed seismic wave field requires a rather large amount of electronic storage space, resulting in huge datasets even for modest problems. Furthermore, these datasets need to be accessed and processed efficiently enough to warrant performing such a simulation in a reasonable time-frame. This calls for specialized storage schemes and high performance hardware, to achieve the level of performance that would make these types of analysis attractive to researchers and practitioners.

There is a need, then, to show that incorporating the spatiotemporal variability of the seismic wave-field is both relevant, in terms of reducing the earthquake loading uncertainty, and feasible with current technology. Furthermore, if coupled with validated wave propagation models, solution methods, and earthquake scenario modeling; this can provide a truly rational means to input earthquake loading for design and performance evaluation of existing and future infrastructure.

1.1. Hypothesis

The main hypothesis in this work is that using a realistic seismic wave as input to a numerical model of site and structure, cæteris paribus, will lead to a system response which is different than that obtained when using an equivalent 1–D site input motion.

1.2. Scope of Study

A numerical model of earthquake source, crust and wave propagation will be used to produce input motions for a three-dimensional model of a structure and site. Initially, several configurations of source geometry will be assessed in order to obtain a case favorable for comparison. To wit, a condition is sought such that simulated three-dimensional response of the site as close as possible to a 1–D site condition, so that a baseline case where the 1–D site assumption is appropriate can be established. Then, a perturbation of this scenario will be explored and the changes in the accuracy of the 1–D site assumption will be evaluated in the presence of soil nonlinearity. The seismic modeling program SW4 (Seismic Wave 4th. order) will be used for earthquake simulations.

Motions will be generated and applied to a finite-element model of the site and structure using the Domain Reduction Method (DRM). The finite-element modeling program Real-ESSI Simulator, will be used for all non-linear finite element simulations. Linear and non-linear finite element models will be used to explore the difference in response in the presence of non-linearity for the fully three-dimensional model.

Then, the free-field site response (e.g. obtained form the simulation without the structure) will be deconvolved in depth using a state-of-practice method to obtain an equivalent 1–D site wave-field. This field will be applied to the same models used before, again using the DRM.

The superstructure displacement and acceleration responses will be computed at all story levels on distinct locations of the floor-plan. The distribution of energy dissipation will be measured and compared in the two cases. Only soil non-linearity will be considered.

Several tools and features of the Real-ESSI Simulator will be implemented or improved in order to achieve the proposed objectives.

The Domain Reduction Method will be re-implemented in Real-ESSI Simulator, with special attention on data access efficiency and simulation in a parallel computing environment.

A new implementation of classical elasto-plastic constitutive model for material response will be presented. This new implementation achieves high-performance by utilizing so-called 'meta-programming' techniques, while providing modularity, extensibility and increased maintainability of the code. Additionally, by maximizing code reuse, the new framework allows the creation of new constitutive models while providing automatic parallelization and other services, so that new model components can be incorporated into the framework without concern about implementation specifics.

The output subsystem of Real-ESSI simulator will be re-written to improve efficient random access to the output datasets. Additionally, the problem of parallel output of Real-ESSI will be addressed.

Finally, a domain-specific language intended to facilitate simulation in Real-ESSI is designed, implemented and presented. This provides a fully-featured programming language and binds finite-element modeling commands to a syntax which mimics natural language and enforces physical unit safety in modeling, among other features.

1.3. Original Features

This dissertation presents a complete non-linear earthquake-soil-structure interaction (ESSI) analysis for a fictitious, although realistic, earthquake source, site and structure. The novelty lies in using a separate dedicated code for simulating an earthquake source by specification of the earthquake scenario, and using this program to generate data for ESSI modeling with DRM. Aspects regarding the requisites for this procedure to be successful are discussed.

Also original to this work, is the use of DRM with realistic earthquake motions and a non-linear representation of the soil medium. A comparison of this 3–D modeling approach with state-of-practice 1–D site modeling yields new insights into the old and new methods.

Additionally, all the preceding is done–and actually made possible–by using parallel computations in the Real-ESSI Simulator program, executed on clusters and shared-memory parallel computers. Special attention has been placed at code scalability at all of the involved program sub-systems. Several opportunities for performance improvement have been identified and seized, while others have been identified for future work. This work, mostly not explicitly stated herein, is the result of the author's years of research at UC Davis and represent a significant step forward in terms of tools available for non-linear ESSI simulations.

A novel framework for elasto-plastic computations based on modern programming techniques and with regards to sensible code design principles is presented, implemented an tested.

Finally, a domain specific language (DSL) for modeling and simulations in ESSI is designed, implemented and presented. This particular piece of work was one of the author's earliest contributions and has reached a high level of maturity and stability. It has been tested extensively and is used now by several other researchers, collaborators and institutions.

1.4. Summary of Contents

Chapter 1 contains the main motivation and central hypothesis overarching this work.

Chapter 2 presents an overview state-of-practice in SSI modeling, and is meant to establish key concepts and terminology used in the main body of this work. The Domain Reduction Method is presented as an attractive method for the future of ESSI simulations and as cornerstone technology enabling this work.

Chapter 3 presents key components of the computational framework used. The new implementation of classical elasto-plasticity using C++ templates is presented and discussed. This new framework exhibits high performance and also adheres to code development standards of code reuse, maintainability, and extensibility. Also presented in this chapter is the new domain specific language for simulation in ESSI. Finally, the redesign of the output subsystem for RealESSI program is presented.

Chapter 4 regards the numerical simulation of earthquake sources and the propagation of seismic motions inside Earth's crust. Key seismological concepts are introduced. Examples are presented which showcase the effect of source geometry and layering on the motions experienced at a site. Furthermore, the effect of a non-homogeneous crust, presenting an intrusion, are presented and discussed. Earthquake scenarios are postulated and selected for modeling in the next phase of research. Once the scenarios are selected, the issue of how to properly couple motions generated in SW4 with RealESSI through the DRM is discussed.

Chapter 5 presents a full ESSI analysis, using linear and non-linear material properties. The constitutive models used and their behavior is discussed. A comparison is done for the cases of 3–D and 1–D incident wave-fields as well as the differences which arise from using a non-linear material.

Chapter 7 summarizes the main findings of this work and identifies future research directions.

Appendix A presents a review of the non-linear finite element method.

Appendix B shows simple examples on using the new DSL for ESSI simulations.

CHAPTER 2

Methods for Earthquake-Soil-Structure Interaction Analysis

2.1. Soil-Structure Interaction Modeling: State-of-Practice

State-of-practice in soil-structure interaction (SSI) modeling for earthquake response might best be represented by the document (NEHRP Consultants Joint Venture (2012)). In this document a summary of the body-of-knowledge is presented on the subject of SSI modeling and simulation. It is not, however, a document that presents the state-of-art on its subject matter. Although technically sound for historical reasons, more rational approaches can now be utilized to evaluate the response of soil-structure systems.

It has been well established that the influence of SSI during earthquake motions, or any other transient excitation for that matter, is not easily evaluated. Very general intuition leads to the reasoning that if the structure is very flexible in comparison to the underlying site materials, it is possible to assume a rigid base (i.e. no SSI modeling) without great loss of precision. Conversely, if the structure is very rigid in comparison to the site, as is usually the case for nuclear power plants (NPPs) and other very stiff structures, SSI is of great importance and cannot be overlooked without incurring in error.

Other situations are not so clear cut but it is safe to say that, in general, fixed base response will differ from response which considers SSI effects. In other words, it is unreasonable to overlook SSI effects in general. Furthermore, the very concept of speaking of 'soil-structure interaction', and derived terms, is a misleading one inasmuch as it gives the analyst the impression that it is a modeling refinement which might be optional or accessory. It is better to think in terms of simply modeling a system with all of the factors that might contribute to its behavior and performance, SSI generally being a prime suspect.

Once the necessity of SSI is established a myriad of approaches are available to the modeler to develop model a soil/structure system. Every modeling endeavor should start from very

simple models which capture only gross behavioral features and increasingly refine the analysis until the most detailed model gives results indistinguishable from those obtained in preceding less-detailed analyses.

SSI Modeling approaches currently used in research and engineering practice in order of increasing complexity are:

- **No SSI**

  In this modeling strategy the structural elements arriving the place of contact with the ground are given a zero-displacement boundary condition. Two approaches can be used then to apply ground motions depending on the size of the structure with respect to the wave-propagation conditions. First, if the extension of the structure is small relative to the wavelength of involved motions, a relative coordinate system can be established at the base and an acceleration record directly applied as body forces proportional to the mass and acceleration at each point in the structure in accordance with Newton's second law of motion (Newton et al. (1729)). Second, if it expected that wave passage effects are important (for example in long, flexible bridges) then the displacement and velocities of the recorded motions must be applied directly at the points of contact between structure and ground.

- **Foundation Stiffness Approach**

  Instead of applying a rigid boundary condition with zero displacements, a flexible support is modeled by using equivalent springs and dashpots (linear dampers) to model soil deformation and energy dissipations. This is the oldest approach available, a review of the methods can be found in Stewart et al. (1999a) and Stewart et al. (1999b), as well as Kramer (1996) and, of course, NEHRP Consultants Joint Venture (2012). Determining stiffness and damping constants is done by fitting experimental data. Some methods propose using frequency-dependent constants to achieve better results, but limit the analysis to the frequency domain, excluding any possibility of modeling structure non-linearity.

  Loading, in this case, is a more delicate subject. In general, the relative coordinate system approach is used with motions prepared to match site conditions.

- **P-Y springs**

  See, for example Boulanger et al. (1999). Basically extends the previous approach to include springs with non-linear behavior. The idea comes from experimental tests in lateral behavior of piles and pile-groups, where a the pile cap is pushed and its deformation and load evaluated to develop a response curve. This can then be fitted to several unidimensional (force-deformation) non-linear hysteretic springs and a pushover analysis carried out to verify. The idea is extended to model lateral response of walls and also strip footings and other foundation types.

  The same loading strategy as before is used to load these types of models.

- **1–D wave propagation models for soil**

  Soil is explicitly modeled as a unidimensional model wherein waves travel into the structure. Elements representing continuous surfaces of soil in shear strain are placed under the structure and a seismic record is input at the base. Depending on whether there is an underlying rock layer, seismic records can be directly input at the base or de-convolved (see Mejia and Dawson (2006)) for an up-to-date review) to depth.

  In general, three-axis motion can be considered as well as soil non-linearity if a suitable model is available. Usually only one component, the horizontal, is considered at a time.

  While more rational than any of the preceding methods, response of the soil is not unidimensional, especially in the vicinity of the foundation. Distributed plasticity effects cannot, therefore, be considered. Additionally, this assumes that earthquake motion propagates vertically in all components, resulting in the fact that the vertical component is realized by P-waves within the 1–D domain.

- **Direct Methods**

  These models explicitly represent the complete soil mass and the structure in three dimensions as well as the soil-structure interface. A continuum mechanics numerical scheme, prominently finite elements or finite differences, is used for soil and possibly parts of the structure. Soil can be modeled linear, linear–equivalent (i.e. Lymser (1988)) or fully non-linear.

Motions are conventionally input using a similar approach to the de-convolution and 1–D site analysis scheme. A record is de-convolved to obtain displacements in depth for the site, and then the wave field is placed at the boundary using a stress condition which allows outward propagation of waves. Out-going motions, or radiation damping, is modeled using some type of absorbent boundary conditions at the domain edges such as Lysmer dashpots. The arguable advantage of this input method is that recorded acceleration motions (accelerograms) can be used directly input into the system. The disadvantage is that, in order to do so, it has to be assumed that the earthquake wave field is essentially 1–D, disregarding surface waves, inclined seismic motions, wave passage effects and assuming that the vertical component is produced by a vertically propagating compression wave.

Conversely, more realistic input methods such as the Domain Reduction Method (DRM, explained in the next section) can be used for a more satisfactory representation of the incident wave field with all of its components. The drawback to this method is that only simulated motions which satisfy the wave equation can be used leading to an increase in computational cost with respect to the '1–D site' methodology.

The most comprehensive model possible for SSI is one that considers the complete continuum and also incorporates the incident seismic wave-field in its full complexity. Therefore, the model includes an explicit specification of an earthquake scenario. Finally, to complete the wave propagation picture, any vibrations caused by the motion of the structure must be radiated out of the domain and damped. The only possibility to achieve this ultimate level of detail is to use a method such as DRM to input motions along with seismologically valid earthquake simulations. This is also the only way to rationally simulate and test pervasive assumptions used in practice such as the '1–D' site assumption or the reduction of seismic demands due to 'base slab averaging' (kinematic interaction) of motions at a rigid base.

Two major reasons have impeded the adoption of the method and its usage for mainstream practice: first the high computational demand in terms of storage that the method requires and, second, the confidence in the ability to correctly represent seismic propagation of motions from a model of the earthquake, through a model of the crust into a model of the site and structure.

Both earthquake modeling and computational power are now at a level where performing this type of analysis routinely is available to researchers practitioners at an accessible computational cost. It becomes important, therefore, to determine when this most-complete methodology should be used, i.e. when it yields results which are different from other methodologies.

2.2. The Domain Reduction Method

The domain reduction method (Bielak et al., 2003; Yoshimura et al., 2003) (DRM) exploits the weak coupling between source rupture dynamics and local site response to separate the ESSI problem into two problems to be solved independently: (i) source and path propagation and (ii) site and structural response. It provides a rational means to input realistic elastic wave fields computed on a 'free-field' domain into a model which contains a feature or perturbation of the free-field. Such features might include a site with softer soil conditions prone to nonlinearity, a structure, or a topographic irregularity to name a few examples.

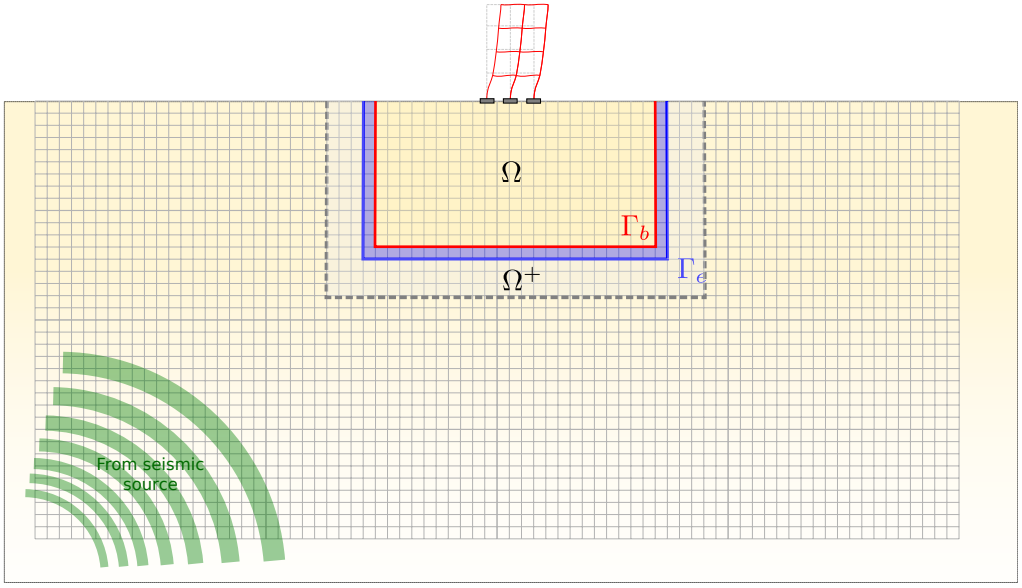


FIGURE 2.1. Complete ESSI problem consists of source and path modeling as well as local site and structural response.

The following is a re-derivation of the method. Figure 2.1 shows a computational domain which includes a source earthquake, geology and a local feature which might be a structure or another type of perturbation from the free-field condition. The domain reduction method starts

by considering the discretized finite-element equations of motions representing the complete problem.

$$\boldsymbol{M}\ddot{\boldsymbol{u}} + \boldsymbol{K}\boldsymbol{u} = \boldsymbol{F}(t) \tag{2.1}$$

A partition of the domain is defined as follows: Γ_b is defined as a surface which encloses the feature or site of interest, Γ_e is the external surface of all finite elements in contact with Γ_b. Γ_b and Γ_e are called the internal and external boundaries respectively. Finite element Nodes located on Γ_b are called 'boundary' nodes and their displacements are collected on the displacement vector $\boldsymbol{u}_b$. Nodes located in the volume of finite elements enclosed by Γ_b and Γ_e but not on Γ_b are called 'exterior' nodes and their displacements stored in the vector $\boldsymbol{u}_e$. Finally, nodes inside the DRM domain (Ω) are called 'interior nodes' and their displacements stored in the vector $\boldsymbol{u}_i$. With this notation in mind, Equation 2.1 can be partitioned as follows.

$$\begin{bmatrix} \boldsymbol{M}_{ii}^{\Omega} & \boldsymbol{M}_{ib}^{\Omega} & \boldsymbol{0} \\ \boldsymbol{M}_{bi}^{\Omega} & \boldsymbol{M}_{bb}^{\Omega} + \boldsymbol{M}_{bb}^{\Omega+} & \boldsymbol{M}_{be}^{\Omega+} \\ \boldsymbol{0} & \boldsymbol{M}_{eb}^{\Omega+} & \boldsymbol{M}_{ee}^{\Omega+} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{u}}_i \\ \ddot{\boldsymbol{u}}_b \\ \ddot{\boldsymbol{u}}_e \end{bmatrix} + \begin{bmatrix} \boldsymbol{K}_{ii}^{\Omega} & \boldsymbol{K}_{ib}^{\Omega} & \boldsymbol{0} \\ \boldsymbol{K}_{bi}^{\Omega} & \boldsymbol{K}_{bb}^{\Omega} + \boldsymbol{K}_{bb}^{\Omega+} & \boldsymbol{K}_{be}^{\Omega+} \\ \boldsymbol{0} & \boldsymbol{K}_{eb}^{\Omega+} & \boldsymbol{K}_{ee}^{\Omega+} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_i \\ \boldsymbol{u}_b \\ \boldsymbol{u}_e \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{P}_e \end{bmatrix} \tag{2.2}$$

Here $\boldsymbol{K}_{**}$ and $\boldsymbol{M}_{**}$ are the blocks of the stiffness and mass matrices which belong to the internal domain only ($** = ii$), boundary layer of nodes ($** = bb$), external domain ω^+ ($** = ee$) or the coupling terms. The boundary completely separates the internal from the external domains, so there is no coupling of the type ($** = ie$). The forcing term P_e will contain the equivalent forces due to the seismic source.

Next, the analysis is restricted to the internal domain and the boundary as shown in Figure 2.2 and the motion at the external and boundary nodes is separated into a background or free-field motion $\boldsymbol{u}_{eo}$ and $\boldsymbol{u}_{bo}$ plus a correction term due to the feature $\boldsymbol{w}_e$ and $\boldsymbol{w}_b$.

$$\boldsymbol{u}_b = \boldsymbol{u}_{b0} + \boldsymbol{w}_b \tag{2.3}$$

$$\boldsymbol{u}_e = \boldsymbol{u}_{e0} + \boldsymbol{w}_e \tag{2.4}$$

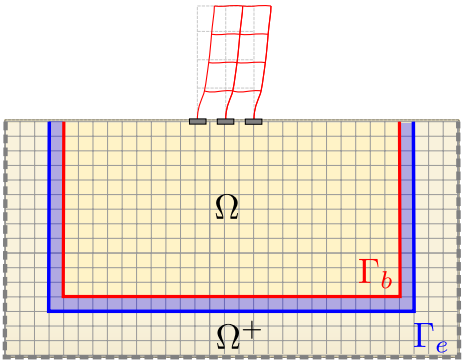


FIGURE 2.2. Complete ESSI problem consists of source and path modeling as well as local site and structural response.

The free-field motion will be obtained in a separate analysis, while the additional correction motion will be due to the second analysis which includes the feature. With this $\boldsymbol{w}_b$ and $\boldsymbol{w}_e$ become the new unknowns. Replacing the equation above into 2.2 and re-arranging terms yields

$$\begin{bmatrix} \boldsymbol{M}_{ii}^{\Omega} & \boldsymbol{M}_{ib}^{\Omega} & \boldsymbol{0} \\ \boldsymbol{M}_{bi}^{\Omega} & \boldsymbol{M}_{bb}^{\Omega} + \boldsymbol{M}_{bb}^{\Omega+} & \boldsymbol{M}_{be}^{\Omega+} \\ \boldsymbol{0} & \boldsymbol{M}_{eb}^{\Omega+} & \boldsymbol{M}_{ee}^{\Omega+} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{u}}_i \\ \ddot{\boldsymbol{u}}_b \\ \ddot{\boldsymbol{w}}_e \end{bmatrix} + \begin{bmatrix} \boldsymbol{K}_{ii}^{\Omega} & \boldsymbol{K}_{ib}^{\Omega} & \boldsymbol{0} \\ \boldsymbol{K}_{bi}^{\Omega} & \boldsymbol{K}_{bb}^{\Omega} + \boldsymbol{K}_{bb}^{\Omega+} & \boldsymbol{K}_{be}^{\Omega+} \\ \boldsymbol{0} & \boldsymbol{K}_{eb}^{\Omega+} & \boldsymbol{K}_{ee}^{\Omega+} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_i \\ \boldsymbol{u}_b \\ \boldsymbol{w}_e \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{P}_e^{\text{eff}} \\ \boldsymbol{P}_b^{\text{eff}} \end{bmatrix} \tag{2.5}$$

Where now the earthquake motion due to the free-field problem has been replaced into effective forces at the boundary $\boldsymbol{P}_b^{\text{eff}}$ and $\boldsymbol{P}_e^{\text{eff}}$. Where

$$\boldsymbol{P}_b^{\text{eff}} = -\boldsymbol{M}_{be}^{\Omega+}\ddot{\boldsymbol{u}}_{e0} - \boldsymbol{K}_{be}^{\Omega+}\boldsymbol{u}_{e0} \tag{2.6}$$

$$\boldsymbol{P}_e^{\text{eff}} = \boldsymbol{M}_{eb}^{\Omega+}\ddot{\boldsymbol{u}}_{b0} + \boldsymbol{K}_{eb}^{\Omega+}\boldsymbol{u}_{b0} \tag{2.7}$$

If the original motions were developed with additional damping, then a damping matrix should also be added analogously.

It is indeed very possible that the second analysis involving a modification of the free-field condition will require a time-step which is smaller than the one used to compute the DRM

motions. This will necessitate the interpolation of the DRM motions for this analysis, and this requires a word of caution.

Because both displacements and accelerations are explicitly needed in the formulation, care must be taken when interpolating so as to provide a consistent interpolation of both displacements and accelerations. Interpolating linearly in accelerations results in a cubic interpolation for displacements. Whereas linear interpolation in displacements results in *zero* interpolated accelerations, which is wrong. Also wrong is to interpolate linearly in both accelerations and displacements, and note that this is equivalent to interpolating the DRM forces directly.

Suppose that data is needed at t^* between time-steps $t_{i-1} < t^* < t_i$ where displacements and accelerations are $(\boldsymbol{d}_{i-1}, \boldsymbol{a}_{i-1})$ and $(\boldsymbol{d}_i, \boldsymbol{a}_i)$ respectively.

Accelerations are interpolated linearly:

$$\boldsymbol{a}^* = \boldsymbol{a}_{i-1} + \frac{\boldsymbol{a}_i - \boldsymbol{a}_{i-i}}{\Delta T}\left(t^* - t_{i-1}\right) \tag{2.8}$$

Where $\Delta T = t_i - t_{i-1}$. Displacements must satisfy a cubic polynommial in t^*

$$\boldsymbol{d}^*(t^*) = A\left(t^* - t_{i-1}\right)^3 + B\left(t^* - t_{i-1}\right)^2 + C\left(t^* - t_{i-1}\right) + D \tag{2.9}$$

Of which the time derivative is

$$\ddot{\boldsymbol{d}}^*(t^*) = \boldsymbol{a}^*(t^*) = 6A\left(t^* - t_{i-1}\right) + 2B \tag{2.10}$$

The coefficients A, B, C, and D must satisfy:

$$\boldsymbol{a}^*(t_{i-1}) = a_{i-1} \tag{2.11}$$

$$\boldsymbol{a}^*(t_i) = a_i \tag{2.12}$$

$$\boldsymbol{d}^*(t_{i-1}) = d_{i-1} \tag{2.13}$$

$$\boldsymbol{d}^*(t_i) = d_i \tag{2.14}$$

By applying these conditions to the polynomial the following coefficients are obtained.

$$A = \frac{\boldsymbol{a}_i - \boldsymbol{a}_{i-1}}{6\Delta T} \qquad B = \frac{\boldsymbol{a}_{i-1}}{2}$$

$$C = \frac{\boldsymbol{d}_i - \boldsymbol{d}_{i-1}}{\Delta T} - \frac{\boldsymbol{a}_i - 2\boldsymbol{a}_{i-1}}{6}\Delta T \qquad D = \boldsymbol{d}_{i-1}$$

The effect of using an alternate interpolation method results in increased interpolation errors. These errors manifest themselves as low-amplitude high-frequency noise which propagates throughout the domain.

2.3. Review of Previous Research on Using DRM for ESSI

(Bielak et al., 2003) presented the original formulation on DRM and then Yoshimura et al. (2003) used 3–D Greens functions to develop seismic motions for the analysis of a basin and a hill and verify the DRM methodology. The same program was used to generate the motions and do the DRM computation, with the advantage being that meshes could be discretized independently for efficiency.

Psarropoulos et al. (2007) used a precursor of the modern DRM method, also by Bielak, to compute linear and non-linear valley amplification effects due to a vertically incident shear wave computed from motion record deconvolutions.

Jeremić et al. (2009) also computed DRM motions from deconvolution of surface acceleration records and to study the earthquake response of a model of a bridge including soil-foundation-structure interaction.

Kontoe et al. (2012) used DRM as input method for a 1–D field developed by record deconvolution to explore the seismic response and interaction of soil retaining systems. 2-D DRM finite element models were used for this purpose.

Tripe et al. (2013) assess the linear topographic amplification response using the DRM. 1–D vertically propagating shear wave motions were developed for this study and input using DRM. Notably, absorbing boundaries where provided using a Lysmer style boundary condition.

Jeremić et al. (2013) computed the response of simplified, modally equivalent nuclear power plant and internal structure models to seismic motions developed using an incoherency model

for seismic waves and inputing them using DRM. NPP model incorporates contact non-linearity at the soil-foundation interface.

Zhong and Huang (2014) use the DRM methodology to verify a simplified method for computing the response of caisson-pile foundations. The input wave field in this case was unidimensional.

Isbiliroglu et al. (2015) used the DRM method to compute the seismic response of building clusters, using simplified linear models of buildings and soil.

CHAPTER 3

Earthquake-Soil-Structure-Interaction Simulation Platform

3.1. Classical Elasto-Plasticity Implemented Using C++11 Template Metaprogramming

3.1.1. Introduction. A classical small deformation elasto-plasticity constitutive model describes the evolution of stress at a material point with changes in material strain. Here the word *classical* refers to traditional elasto-plastic constitutive models composed of: (i) a yield surface, (ii) an elasticiy law, (iii) a plastic flow direction (or plastic potential function), and (iv) a hardening law for the model's internal state variables, and formulated in terms of stress and strain rates. This section presents an efficient implementation of classical elasto-plasticity algorithms using the template meta-programming features of C++ programming language (C++11 to be precise), which also allows code modularity and reuse while also providing an efficient implementation.

Classical elasto-plasticity is a well understood framework for developing constitutive models for for many material types. This encompasses a large class of models, with the capability of modeling many of the observed behaviors of materials. Excluded from this representation are visco-elastoplasticity models, boundary surface plasticity models, multi-yield-surface models, and such. Notwithstanding, the implementation concepts developed herein can be extended to these types of models by introducing the additional abstractions.

The framework for the solution of the constitutive rate equations is independent of the actual specification of the different components. In other words, one need not specify the actual yield function, elasticity law, plastic-flow direction and hardening laws in order to develop an algorithm that advances the state of a classical elasto-plastic model. Because of this, solution algorithms can be developed independently from the model components, allowing for core modularity and reuse.

Specific instances of the model components can be independently developed and then a particular selection of them combined to produce a unique material model. This means that

with only a few particular implementations of the components, combinatorially-many material models can be instantiated. For example, if two of each components are available, with four components for classical elasto-plasticity, these can be combined to produce $2 \times 2 \times 2 \times 2 = 16$ distinct material models.

A modular code which separates solution procedure (and other services), provided that the design is sensible, makes it very easy to maintain and extend. For example, if an opportunity for better performance is discovered in the integration algorithm (a maintenance issue) it can be implemented by modifying only that part of the code. Furthermore, new solution procedures (for example, exhibiting different interesting properties such as convergence rates and stability) can be added in one place (an extensibility issue) and become available to all the material models at once. This is ideal from the software developer and maintainer point of view, which might elect to meet these requirements by object-oriented (OO) software engineering principles[1].

An OO design will assign all components and the main material integration procedures to objects and the functionality will be implemented by the interaction of these objects. Menbtrey et al. (1993) implemented a rudimentary version of these concepts by applying OO-design to a finite-element code and, in particular, to material models restricted to J_2 plasticity. Zabaras and Srikanth (1999) used an OO-design to perform computations in two-dimensional large-deformation elasto-plasticity, polymorphism was used on complete models which were called by a constitutive integrator driver class. Jeremić and Yang (2002) showed that a full-blown OO-design is possible for elasto-plastic computations by further abstracting the model components with polymorphic classes and also embedding them in a driver class to generate a material model.

Despite its beauty and advantages, OO design as above has relied on run-time polymorphism at different levels. This has the main disadvantage that the Runtime Type Information (RTTI) is used at runtime to determine which component instance gets called. Furthermore, because the types of the components are not known at compile-time, the compiler cannot do useful optimizations like function inlining, object-pointer indirection, to name a few. This incurs in

[1] which can be applied to *any* language, not just those with OO features.

an overhead each time the functions are needed. Since these functions are used in the lowest loop-levels of typical finite-element codes, avoiding this performance hit is crucial.

As noted by Cecilio, Lira et al. (2010), the use of template meta-programming on the material class, declaring the model components as template parameters, avoids these issues altogether by allowing compile-time determination of data types and enabling the corresponding optimizations. Their design choice, however, was not truly polymorphic at the component level since all instances of the components must define all their functions to work, unless polymorphism is used on the base component classes, which defeats the purpose of the approach. This limits the code re-usability and extensibility for that particular implementation, leaving the burden of a complete implementation to each implementer of model components. This hinders flexibility and extensibility of the code.

This article uses template meta-programming to achieve compile-time polymorphism, allowing to truly leverage compiler optimizations of operations on known types, while at the same time provides a framework for extension of the code. This, in conjunction with an efficient tensor data class, provides maximum flexibility of polymorphic object-oriented code design, while retaining the efficiency of static-typing.

It must be noted that ultimate performance can only be achieved by exploiting the particular mathematical form of the model components, and following through with the computations by hand. This must be done for each combination of the models, or at least those of interest, which can be cumbersome and error prone.

Section 3.1.2 reviews the basics of classical elasto-plasticity necessary for a minimum implementation. Next, Section 3.1.3 lists the template-metaprogramming design patterns used to achieve the implementation and the rationale behind them. Section explains the design of the abstract base-classes used to create new materials. Section 3.1.6 exemplifies how to use the design to instantiate a Von-Mises and Drucker-Prager material models, both with isotropic and kinematic linear hardening rules. Finally, Section 3.1.7 uses the generated materials to compute response for simple strain-controlled cases.

3.1.2. Components of classical elasto-plasticity. A classical small deformation elasto-plasticity constitutive model is used to describe the incremental behavior of material stress when it undergoes an increment in strain. It is defined by the system of non-linear ordinary differential equations

$$\mathrm{d}\sigma_{ij} = E^{ep}_{ijkl}(\boldsymbol{\sigma}, \boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q_N})\,\mathrm{d}\epsilon_{kl} \tag{3.1}$$

Where $\mathrm{d}\epsilon_{ij}$ is the increment in strain which usually comes from a displacement increment obtained from one step of a global finite-element analysis. $\mathrm{d}\sigma_{ij}$ is the increment in stress due to the increment in strain and is the sough results. The elasto-plastic modulus tensor $E^{ep}_{ijkl}(\boldsymbol{\sigma}, \boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q_N})$ linearly relates stresses and strains within one increment, and is a function of current stress and a set of N internal variables $\{\boldsymbol{q}_i\}_{i=1}^{N}$. These internal variables might be scalars, second-order tensors or fourth order tensors.

Next, the strain increment is separated in its elastic and plastic components.

$$\mathrm{d}\epsilon_{ij} = \mathrm{d}\epsilon^{e}_{ij} + \mathrm{d}\epsilon^{p}_{ij} \tag{3.2}$$

The main unknown here is the increment in plastic deformation $\mathrm{d}\epsilon^{p}_{ij}$. Equation 3.1 can be expressed solely in terms of the elastic strain increment (cfr. Crisfield (1991))

$$\mathrm{d}\sigma_{ij} = E_{ijkl}(\boldsymbol{\sigma})\,\mathrm{d}\epsilon^{e}_{kl} \tag{3.3}$$

Where now the elastic modulus tensor $E_{ijkl}(\boldsymbol{\sigma})$ only depends on current state of stress and the elastic strain increment is known once the plastic strain increment is known $\mathrm{d}\epsilon^{e}_{kl} = \mathrm{d}\epsilon_{kl} - \mathrm{d}\epsilon^{p}_{kl}$. This elasticity law is one of the components of classical elasto-plasticity. An additional three additional components are needed to determine the plastic-strain increment. The components are:

- Elasticity law, $E_{ijkl}(\boldsymbol{\sigma})$.
- Yield surface, $f(\boldsymbol{\sigma})$.
- Plastic flow direction, $\boldsymbol{m}$.
- Hardening law for the internal variables.

The yield surface, $f(\boldsymbol{\sigma})$, is a scalar function of the stress tensor which evaluates to a negative number when the material is elastic. The locus of the stress points where the function is equal to zero defines the yield surface. Within it, the response is possibly non-linear but energy-conservative ($\mathrm{d}\epsilon^p_{ij} = 0$). Stress states which evaluate to a positive value of the yield function are forbidden. In addition to being a function of stress state, the yield function is also parametrized by the internal variables of the constitutive model which control its size, location in stress space and shape. Moving, resizing and reshaping of the yield surface allows attaining states of stress previously unavailable at the cost of incurring in elasto-plastic deformations.

The plastic-flow direction $\boldsymbol{m}$ is a unit-norm tensor which defines the direction of plastic strain for a given increment in stress through. Together with the plastic multiplier $\mathrm{d}\lambda$ the plastic-strain tensor is defined to be

$$\mathrm{d}\epsilon^p_{ij} = \mathrm{d}\lambda m_{ij} \tag{3.4}$$

This makes the plastic-multiplier $\mathrm{d}\lambda$ (a positive scalar) the primary unknown in this setup. The plastic-flow direction might depend on the current state of stress and the state of the internal variables. Hypo-plastic models also make it depend on the direction of loading through the strain increment or the plastic-strain increment, possibly resulting in incrementally non-linear models. An important family of plastic-flow directions are those which can be derived from a potential function defined in the stress-internal variable-space $Q(\sigma, \boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_N)$, through the derivative

$$m_{ij} = \frac{\partial Q}{\partial \sigma_{ij}} \tag{3.5}$$

If the plastic-potential function $Q()$ is selected equal to the yield function $f()$, the resulting model is called *associative* plasticity model, otherwise the model is *non-associative*.

Since the plastic multiplier defines the magnitude of the plastic strain increment, then in the case of elastic response $f() < 0$ it must be zero, otherwise it is positive. The whole setup can be summarized in the conditions.

$$
\begin{aligned}
f(\boldsymbol{\sigma}, \boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_N) &\leq 0 \\
\mathrm{d}\lambda &\geq 0 \\
f(\boldsymbol{\sigma}, \boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_N)\, \mathrm{d}\lambda &= 0
\end{aligned}
\tag{3.6}
$$

Finally, the hardening law(s) describes how to evolve or update the internal variables once the plastic multiplier is known. They take the form:

$$
\mathrm{d}\boldsymbol{q}_i = \mathrm{d}\lambda \boldsymbol{h}_i(\boldsymbol{\sigma}, \boldsymbol{q}_i) \tag{3.7}
$$

Which is the evolution law for the i-th internal variable.

Thus far, no attempts have been made at the specification of the particular expressions of the components. This will be explored in Section 3.1.6 where the models are specialized. For now, it is assumed that all these components are defined with the properties discussed.

Additionally, in this work, the LTensor library by Limache and Rojas Fredini (2008) will be used to represent and store tensor variables. This library allows expressiveness in C++ code dealing with tensors as it allows using indicial notation directly in code[2]. The author believes that its use will not only clarify the presented code, but also appeal to readers with a background in continuum mechanics.

3.1.3. Template meta-programming techniques. An exhaustive treatment of templates and meta-programming techniques is beyond the scope of this work. The reader is referred to Abrahams and Gurtovoy (2004) for a excellent introduction.

In the C++ language a template is a language feature that allows writing code independent of the data-types involved. For example, using templates it is possible to write a the following function:

```
1 template<typename T> sum(T a, T b) { return a+b; }
```

[2]This is a successful instance of C++ template metaprogramming

The template parameter `T` can be replaced with any type that supports the binary addition operation by invoking the function and replacing the type, which is called template *specialization*. For example, specializing for `double` type:

```
double pi=3.14, e=2.71;
double s = sum<double>(pi, e);
```

This allows writing certain kinds of algorithms only once and then *instantiating* them to different data-types.

Two template-based techniques will be reviewed here. Static polymorphism using the 'Curiously-Recurring Template Pattern' and the use of variadic templates to generate containers of arbitrary types. This last technique will be used to implement a way to access an operate on an arbitrary number of internal variables in the elasto-plastic model.

3.1.3.1. *Static polymorphism with the 'Curiously-Recurring Template Pattern'.* Polymorphic objects are those that have at least one member function declared `virtual`, that is, overloadable by derived classes. This concept is at the base of object-oriented design. For example, consider the classes:

```
class Material
{
  public:
  virtual int integrate() = 0;
}

class Sand : public Material
{
  public:
  int integrate()
  {
    //Implementation for sand
  }
}
```

Similarly other materials such as 'Clay', 'Rock' and 'Metal' might define their own integration functions. Polymorphism, in the context of constitutive models, can be (and is) used to store in memory pointers to material models at element Gauss-points. In order to do constitutive integration this list of pointers is transversed and for each pointer an `integrate()` member function might be invoked. Since the `integrate()` member function is called from a pointer to

an object of class `Material`, then the correct implementation of `integrate()` must be determined in runtime. This is called dynamic-dispatch, and it incurrs in the penalty of determining the real type of the pointer as well as the pointer de-reference once the type is determined.

In C++ template meta-programming 'Curiously-Recurring Template Pattern' (CRTP) is a design idiom which simulates compile-time polymorphism. The idiom involves deriving a class from a template base-class using the derived class as template parameter. This allows calling functions defined in the derived-class from the base class, providing a mechanism for function overload. Re-writing the example above in terms of the CRTP is done as follows.

```
1 template<class T>
2 class Material {};
3
4 class Sand : public Material<Sand> {};
```

Note that here `Sand` is not a template class, it specializes the `Material` class which is a template. To simulate polymorphism, the idiom applied to the example is:

```
1  template<class T>
2  class Material
3  {
4    public:
5    int integrate()
6    {
7    return static_cast<T*>(this)->integrate();
8    }
9  };
10
11 class Sand : public Material<Sand>
12 {
13   public:
14   int integrate()
15   {
16
17   }
18 }
```

If `integrate()` is invoked though a pointer to the base class (`Material`) then this pointer is up-casted to the appropriate type (in compile-time) and the derived-class function invoked without the overhead of dynamic-dispatch and pointer dereferencing.

The CRTP will be used herein as a basic framework to create material models which have the advantages of polymorphism and simultaneously can be heavily optimized by the compiler.

3.1.3.2. *A fixed-length container class for heterogeneous types using variadic templates.* Variadic templates were introduced in in the 2011 version of the C++ standard (C++11), and allow variable number of template parameters. The usage of variadic templates usually involves compile-time recursion to access each of the instantiated template data-types.

Shown here is a simple use-case of variadic templates to create a fixed-length (length is defined by number of template arguments) container class for heterogeneous types with the capability to print the array[3].

```
template <class... Qs> struct HeterogeneousVector
{
  // Base recursion case. Does nothing.
  void print()
  {
  };
};

template <class Q, class... Qs>
struct HeterogeneousVector<Q, Qs...> :
  HeterogeneousVector<Qs...>
{
  HeterogeneousVector(Q q, Qs... qs) :
  HeterogeneousVector<Qs...>(qs...),
  q_i(q) {};

  void print()
  {
  cout << q_i << endl;
  static_cast<HeterogeneousVector<Qs...>*>(this)->print();
  }
  Q q_i;
};
```

The template parameters can be of any type as long as the type supports the stream insertion operator '<<'. The following code creates an array which holds an `int`, a `double` and a `std::string` and prints the array.

```
int main(void)
{
```

[3]Adapted from Eli Bendersky's website `http://eli.thegreenplace.net/2014/variadic-templates-in-c/`, last accessed Sunday January 24th, 2016.

```
  HeterogeneousVector<int, double, string> v(10, 3.14, string("↩
     Hello."));
  v.print();
  return 0;
}
```

The output of this code is

Command line output

```
10
3.14
Hello.
```

The final class of type `HeterogeneousVector<int, double, string>` consists of a chain of classes which inherit form each ither in sequential manner. This is illustrated in the pseudo-C++ code below

```
template<int, double, string>
struct HeterogeneousVector : HeterogeneousVector<double, string>
{
  //...
  int q_i; // Holds the int
};

template<double, string>
struct HeterogeneousVector : HeterogeneousVector< string>
{
  //...
  double q_i; // Holds the double
};

template<string>
struct HeterogeneousVector : HeterogeneousVector<>
{
  //...
  string q_i; //Holds the string
};
```

In order to access a specific element, the container must be casted into a pointer which has removed all data-types up-to and including the type of interest, retaining the rest. The typical case of applying a function to all elements of the container is achieved by a casting process which is done recursively inside the function call. Finally, a base case that does nothing must be implemented to stop recursion at the empty container.

This idiom will be used to invoke operations to an arbitrary number of internal variables. Allowing for models with any number of such variables.

3.1.3.3. *Trait classes.* An example will serve best to illustrate the concept. In plasticity, several algorithms are available to integrate the equations of elastoplasticity. Explicit algorithms are the most straightforward to implement, with the drawback that careful consideration of stability must be made when integrating them. Implicit algorithms offer greater stability at the cost of requiring higher-order derivatives of the yield function to implement.

For a general-purpose implementation the implementer of yield function objects may or may not provide the higher order derivatives necessary to do implicit integration (analytical derivation and casting into a numerically stable form is often a challenging task). It is desirable to be able to disable the compilation of general implicit integration if the higher-derivative expressions are not provided.

There are many ways to implement the idea in the preceding paragraphs. The use of the 'trait-class' idiom is propose here to conditionally compile the appropriate algorithm.

> *Think of a trait as a small object whose main purpose is to carry information used by another object or algorithm to determine "policy" or "implementation details".* –Bjarne Stroustrup

From what was discussed, there are two possible implementation of an 'implicit integration' function, `implicit_integration()`. The function `implicit_integration()` takes a strain increment second-order tensor [4] and updates all internal variables using the best method available returning an error flag. The two implementations are:

```
1  // Don't do implicit integration  [default]
2  int
3  Material::implicit_integration(const DTensor2& strain_increment)
4  {
5    cerr << "Implicit integration unavailable for this material" << ↩
       endl;
6    return -1;
7  }
8
9  // Use implicit-differentiation [if derivatives are provided]
10 int
```

[4] In LTensor library these are types called `DTensor2` for double precision 2nd. order tensor.

```
Material::implicit_integration(const DTensor2& strain_increment)
{
  return
  implicit_integration_implementation(strain_increment);
}
```

Both these function cannot be compile together as they have the same signature, so some method for selecting the appropriate one in compile time is needed. To select between them the `std::enable_if` template:

```
// Don't do implicit integration  [default]
template <typename U = T>
typename std::enable_if<!provides_high_order_derivatives<U>::value, int>::type
Material::implicit_integration(const DTensor2& strain_increment)
{
  cerr << "Implicit integration unavailable for this material" << endl;
  return -1;
}

// Use implicit-differentiation [if derivatives are provided]
template <typename U = T>
typename std::enable_if<provides_high_order_derivatives<U>::value, int>::type
Material::implicit_integration(const DTensor2& strain_increment)
{
  return
  implicit_integration_implementation(strain_increment);
}
```

Notice that the return type of the function (`int`) is replaced by the `enable_if<>` template object. If the template struct `provides_high_order_derivatives<U>::value` evaluates to `true` then the return-type of the function is `int` and the function is enabled, otherwise the return-type is `void` and the function disabled.

Then the definition of `provides_high_order_derivatives<U>::value` is provided

```
template< typename U >
struct provides_high_order_derivatives
{
    static const bool value = false;
};
```

which evaluates to `false` for all `U`. So, if a particular yield function object implementation, say `DruckerPrager_YF`, does indeed provide high-order derivatives (e.g.through a set of function calls), then it can specialize the above template struct for that particular type:

```
1 template< >
2 struct provides_high_order_derivatives<DruckerPrager_YF>
3 {
4     static const bool value = true;
5 };
```

which activates the second definition of the integrator function.

In summary, trait classes are small objects which carry information (traits) about other classes that can be used to control compile-time inclusion of functions, as well as other uses.

3.1.4. Testing Static Polymorphism vs. Dynamic Polymorphism. In order to test the assumption that using CRTP as opposed to RTTI-based dynamic polymorphism yields more optimized code, and to obtain an idea of the performance increases one can expect when comparing both approaches, a simplified example will be used. In this example, different versions of a function that takes a second-order tensor (`DTensor2`) and returns a real number (`double`), called `function_to_compute`, is tested many times within a loop to measure time taken. The function is accessed through a base-class which wraps the function call within an over-loadable function. The wrapper classes are implemented using the CRTP and virtual functions. This scenario is exactly the case that occurs during the evaluation of the yield function, which is usually evaluated several times during integration of the constitutive model making it an important optimization target.

Table 3.1 lists all the versions of `function_to_compute` used to evaluate the performance of the technique. Note that cases 2 and 4 are the same function computed in different ways. Case 6 is an implementation of the evaluation of the Von-Mises yield function, and represents the most realistic case. The different implementations of the function are tested against all available compiler optimization flags to measure also the effect of optimization.

On page 32, the implementations of the wrapper classes and the actual test using both CRTP and virtual functions are shown side by side for comparison.

TABLE 3.1. Definitions of `double function_to_compute(const DTensor2 & sigma)` for each case.

Case	function_to_compute()
1	`return sigma(0,2);`
2	`return sigma(i,i);`
3	`return sqrt(sigma(i,j)*sigma(i,j));`
4	`return sigma(0,0) + sigma(1,1) + sigma(2,2);`

5	`double val = 0;`
	`for (int i = 0; i< 3; i++)`
	`  for (int j = 0; j < 3; j++)`
	`  {`
	`  val += sigma(i,j);`
	`  }`
	`  return val;`

6	`static DTensor2 s(3,3,0);`
	`double p = sigma(0,0) + sigma(1,1) + sigma(2,2);`
	`s(i,j) = sigma(i,j);`
	`s(0,0) -= p; s(1,1) -= p;`
	`s(2,2) -= p;`
	`return sqrt(s(i,j) * s(i,j));`

The test code was compiled on an Ubuntu 14.04.3 laptop running an Intel(R) Core(TM) `i7-4790K CPU @ 4.00GHz`, using `gcc 4.9.3` with the `-std=c++11` flag enabled. 10,000 million function calls were done in all cases except case 6 which was run 1,000 million times. Times were measured using GNU `time` function, accurate within 10 ms. CPU frequency scaling was disabled during test, which removes the effect of CPU throttling.

Dynamic Polymorphism (virtual functions)

```cpp
#include <iostream>
#include <cmath>
#include "ltensor/LTensor.h"
#include "globals.h"

struct BaseDynamic
{
  virtual double f(const DTensor2& sigma)
    = 0;
};

struct DerivedDynamic:
    public BaseDynamic
{
  virtual double f(const DTensor2& sigma)
    return function_to_compute(sigma);
};

int main(void)
{
  DTensor2 sigma(3, 3, 0);
  sigma(0, 2) = sigma(2, 0) = 0.1;
  sigma(0, 0) = 1;

  BaseDynamic* base_ptr =
      new DerivedDynamic();
  double val = 0;
  for (long int i = 0; i<number_of_times; i++)
  {
    val = base_ptr->f(sigma);
  }
  std::cout << "val = " << val << std::endl;
}
```

Static Polymorphism (CRTP)

```cpp
#include <iostream>
#include <cmath>
#include "ltensor/LTensor.h"
#include "globals.h"

template <class T> struct BaseStatic
{
  double f(const DTensor2& sigma)
    return static_cast<T*>(this)->f(sigma);
};

struct DerivedStatic:
    public BaseStatic<DerivedStatic>
{
  double f(const DTensor2& sigma)
    return function_to_compute(sigma);
};

int main(void)
{
  DTensor2 sigma(3, 3, 0);
  sigma(0, 2) = sigma(2, 0) = 0.1;
  sigma(0, 0) = 1;

  BaseStatic<DerivedStatic>* base_ptr =
      new DerivedStatic();
  double val = 0;
  for (long int i = 0; i<number_of_times; i++)
  {
    val = base_ptr->f(sigma);
  }
  std::cout << "val = " << val << std::endl;
}
```

Results of runs for different optimization levels are shown in Table 3.2. It is seen that the technique is advantageous for all optimization levels above `-O0`, which is no optimization. For full optimization, in cases 1, 4, and 5 the performance increase makes the wall-time for CRTP-based function calls drop below the measurement precision. Cases 1 and 2 are essentially the same trace operation with different implementations, yielding a performance hit for Case 2. Case 2 implements the trace using the facilities of the LTensor library, while Case 1 is a direct access implementation. Case 5 and, especially, case 6 are the ones most similar to that of evaluating a yield function. As can be seen, the worst case scenario improvement in speed is 1.6% for CRTP for this function, a worthwhile optimization given that a function call such as this is executed deep within several nested loops. It is expected that this can be further improved by careful implementation of the yield function kernel.

3.1.5. Design of the abstract base classes for classical elasto-plasticity. Figure 3.1 shows the basic relationship between the base classes in this design. The class responsible for implementing core algorithms is called `ClassicElastoplasticMaterial`. It is a template class with the components of elasto-plasticity being template parameters and an additional template parameter `T` meant to be used for CRTP on the specialization of this base class.

Some of the responsibilities of `ClassicElastoplasticMaterial` include:

- Connecting all components into one material model.
- Storing state of stress, strain and plastic strain and 'memory' versions of these variables.
- Provide constitutive integration algorithms acting on components.
- Compute and provide tangent modular matrix.
- Communication, in parallel-computing, with remote versions of this class.
- Implement a material for use in RealESSI (must implement the `NDMaterialLT` interface).

Below, the proposed implementation of the main components of classical elastoplasticity are shown. `ElasticityBase` defines a base class for the response within the yield surface. The class `YieldFunctionBase` provides an interface for the definition of new yield functions. Finally, `PlasticFlowBase` defines a similar interface for the plastic-flow direction. All these classes are class-templates which use the CRTP to forward the member-function calls to the derived components.

TABLE 3.2. Relative speedup of using Static Polymorphism enabled through CRTP vs. Dynamic Polymorphism for different compiler optimization levels.

Optimization level -O3			
Case	T_{SP} (s)	T_{DP} (s)	Speedup
1	0.00*	0.69	100.00%
2	2.13	2.05	-3.90%
3	6.74	8.05	16.27%
4	0.00*	1.07	100.00%
5	0.00*	2.41	100.00%
6	1.75	1.78	1.68%

Optimization level -O2			
Case	T_{SP} (s)	T_{DP} (s)	Speedup
1	0.00*	0.69	100.00%
2	1.99	2.08	4.32%
3	7.91	8.17	3.18%
4	0.00*	1.44	100.00%
5	0.00*	5.23	100.00%
6	1.62	1.74	6.89%

Optimization level -O1			
Case	T_{SP} (s)	T_{DP} (s)	Speedup
1	0.27	2.52	89.28%
2	0.24	3.78	93.65%
3	8.98	9.92	9.47%
4	0.25	2.29	89.08%
5	0.23	7.79	97.04%
6	2.10	2.19	4.10%

Optimization level -O0			
Case	T_{SP} (s)	T_{DP} (s)	Speedup
1	13.31	10.40	-27.98%
2	32.50	30.16	-7.75%
3	309.63	308.96	-0.21%
4	22.15	19.15	-15.66%
5	64.67	59.73	-8.27%
6	60.10	59.85	-0.41%

* Means that actual run-time is insignificant at available precision.

```
1  template <class T> class ElasticityBase
2  { public:
3    ElasticityBase() { }
4
5    DTensor4& operator()(const DTensor2& stress)
6    {return static_cast<T*>(this)->operator()(stress);}
7  };
8
9  template <class T> class YieldFunctionBase
10 { public:
11   YieldFunctionBase(){}
12
13   double operator()( const DTensor2& sigma) const
```

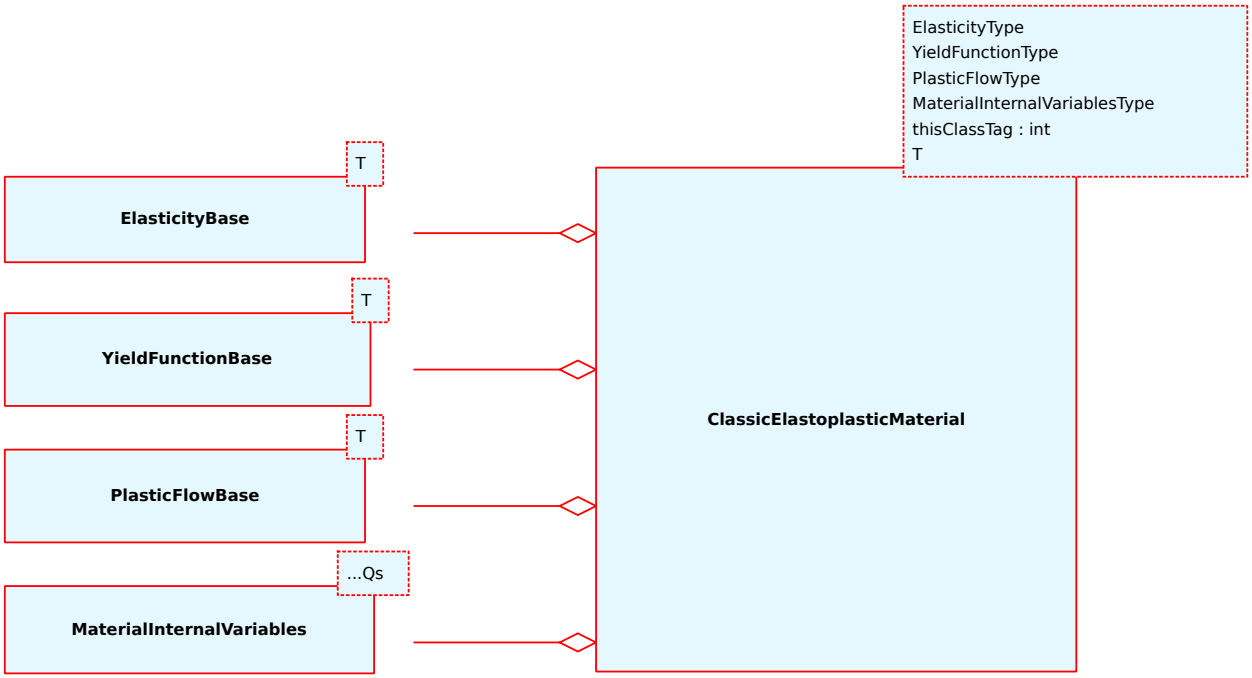


FIGURE 3.1. A UML class diagram for the relationships between the main classes in the framework.

```
14    {return static_cast<T*>(this)->operator()(sigma);}
15
16    const DTensor2& df_dsigma_ij(const DTensor2& sigma)
17    {return static_cast<T*>(this)->df_dsigma_ij(sigma);}
18
19    double xi_star_h_star(const DTensor2& depsilon,
20      const DTensor2& depsilon_pl,
21      const DTensor2& sigma)
22    {return
23     static_cast<T*>(this)->df_dxi_star_h_star(depsilon, depsilon_pl↩
          , sigma);}
24  };
25
26  template <class T> class PlasticFlowBase
27  { public:
28    PlasticFlowBase() {}
29
30    const DTensor2& operator()
31      (const DTensor2 &depsilon, const DTensor2& sigma)
32    {return static_cast<T*>(this)->operator()( depsilon,  sigma);}
33  };
```

Specializations of the components use the CRTP and derive from these classes, passing themselves as template parameter. Then, they implement the required functions and any additional

ones. They also store whatever parameters they might need and provide an appropriate constructor.

Next, a base class is provided for the internal variables of the material model. In this design, the internal variables are responsible of hardening, by defining a hardening function. They store the current and last state and control storage and restoring of the variable state. Finally, it is in charge expressing the requirement that the derived variable must implement sending and receiving the variable for parallel processing.

The design of this base class uses the CRTP to provide polymorphism on the hardening, and delegates the implementation to the derived classes. CRTP is also used in the sending and receiving functions, as the actual implementation will depend on the variable type.

```
template<class VarType, class T>
class EvolvingVariable
{public:

  EvolvingVariable(VarType a_): a(a_), a_committed(a_) {}

  const VarType &getDerivative
                (const DTensor2 &depsilon,
                 const DTensor2 &m,
                 const DTensor2& sigma) const
  {return
    static_cast<const T*>(this)->
      getDerivative(depsilon,  m,  sigma);};

  void evolve(double dlambda,
        const DTensor2& depsilon,
        const DTensor2& m,
        const DTensor2& sigma)
  {const VarType& h = getDerivative(depsilon, m, sigma);
    tmp = h;
    tmp *= dlambda;  //Base type must define "operator*"
    a +=  tmp;       //Base type must define "operator+"
  }

  const VarType& getVariableConstReference() const;
  const VarType& getCommittedVariableConstReference() const;
  void setVar(VarType& v);
  void setCommittedVar(VarType& v);
  void commit()        { a_committed = a; }
  void revert()        { a = a_committed; }
  int sendSelf(int commitTag, Channel &theChannel)
  {return static_cast<T*>(this)->sendSelf(commitTag, theChannel);}
```

```
33    int receiveSelf(int commitTag, Channel &theChannel, ↩
         FEM_ObjectBroker &theBroker)
34    {return static_cast<T*>(this)->receiveSelf(commitTag, theChannel↩
         , theBroker);}
35
36  private:
37    VarType a;
38    VarType a_committed;
39    static VarType tmp;
40  };
```

The base material class, `ClassicElastoplasticMaterial`, does not directly store any variables of type `EvolvingVariable`. Instead, this class hold as data member a list of internal variable references implemented as a `MaterialInternalVariables` class though which it obtains access to perform operations on an arbitrary number of internal variables. This class is implemented as variadic-template class, like the one in the example in the preceding section, that is a container of references to all the internal variables of the model. It has the same interface as `EvolvingVariable`, except it is in charge of forwarding the function calls to each internal variable.

Its responsibilities include setting the values for the variables, setting and restoring variables states and sending and receiving the variables during parallel execution.

```
1  //Base case to end recursion
2  template <class... Qs> struct MaterialInternalVariables
3  {
4    void evolve(double dlambda,
5        const DTensor2& depsilon,
6        const DTensor2& m,
7        const DTensor2& sigma)
8    {}
9
10   void setVars(const MaterialInternalVariables<>& vars) {}
11   void commit() {}
12   void revert() {}
13 };
14
15 //Recurse over variadic template arguments
16 template <class Q, class... Qs>
17 struct MaterialInternalVariables<Q, Qs...> : ↩
      MaterialInternalVariables<Qs...>
18 {
19   MaterialInternalVariables(Q &q, Qs&... qs) : ↩
        MaterialInternalVariables<Qs...>(qs...), q_i(q) {}
```

```
20
21    void setVars(const MaterialInternalVariables<Q, Qs...>& vars)
22    {   q_i = vars.q_i;
23      const MaterialInternalVariables<Qs...>* morevars = static_cast↩
          <const MaterialInternalVariables<Qs...>*>(&vars);
24
25      static_cast<MaterialInternalVariables<Qs...>*>(this)->setVars↩
          (*morevars);
26    }
27
28    void evolve(double dlambda,
29                const DTensor2& depsilon,
30                const DTensor2& m,
31                const DTensor2& sigma)
32    {   q_i.evolve( dlambda,  depsilon,  m,  sigma);
33      static_cast<MaterialInternalVariables<Qs...>*>(this)
34      ->evolve(dlambda,  depsilon,  m,  sigma);}
35
36    void commit()
37    {   q_i.commit();
38      static_cast<MaterialInternalVariables<Qs...>*>(this)->commit()↩
          ;}
39
40    void revert()
41    {   q_i.revert();
42      static_cast<MaterialInternalVariables<Qs...>*>(this)->revert()↩
          ;}
43
44    Q& q_i; //Hold a reference to the actual variable
45  };
```

The implementation of the `ClassicElastoplasticMaterial` base class is left out of the current discussion for space considerations and because its implementation is rather trivial. In a nutshell it uses the CRTP only to call appropriate function overloads (for integration and other services) depending on the defined trait classes of the sub-components. The rest is just utilizing the components with direct calls.

With all this infrastructure, it is now possible to specialize each of these base classes into instances of the model components and come up with new material models. Doing so involves deriving from each component base class, using the CRTP idiom and associating the few function calls that need be done with the correct data.

3.1.6. Instantiation of material components. This section presents the mathematical description of components of elasto-plasticity as well as their implementation in the presented framework. The next section will show how to generate join these components into material models and test them.

3.1.6.1. *Yield functions and plastic-flow directions.* Presented are two yield function models (and their respective associate plastic flow rules) within the J_2 plasticity family of models, namely the Von-Mises and Drucker-Prager models. Both models have two internal variables: the backstress α_{ij}, a second order tensor which specifies the center of the yield surface in stress-space (kinematic hardening); and the scalar strength-parameter k which controls the size of the yield surface (isotropic hardening).

The Von-Mises yield function is a pressure-independent yield function which locus is that of a cylinder in principal stress space, aligned with the mean stress axis. The expression for the Von-Mises yield function is:

$$f(\boldsymbol{\sigma}) = \sqrt{(s_{ij} - \alpha_{ij})(s_{ij} - \alpha_{ij})} - \sqrt{\frac{2}{3}}k = 0 \tag{3.8}$$

Where s_{ij} is the deviatoric component of the stress tensor, $s_{ij} = \sigma_{ij} + p\delta_{ij}$, and $p = -\sigma_{kk}/3$ is the mean stress. Relevant derivatives for this function are:

$$\frac{\partial f}{\partial \sigma_{ij}} = \frac{s_{ij} - \alpha_{ij}}{\sqrt{(s_{pq} - \alpha_{pq})(s_{pq} - \alpha_{pq})}} \tag{3.9}$$

$$\frac{\partial f}{\partial \alpha_{ij}} = \frac{-s_{ij} + \alpha_{ij}}{\sqrt{(s_{pq} - \alpha_{pq})(s_{pq} - \alpha_{pq})}} \tag{3.10}$$

$$\frac{\partial f}{\partial k} = -\sqrt{\frac{2}{3}} \tag{3.11}$$

The Drucker-Prager yield function is a pressure-dependent yield function which locus is a cone defined for positive values of p (compression) and aligned with the mean stress axis. The expression defining the Drucker-Prager yield function is:

$$f(\boldsymbol{\sigma}) = \sqrt{(s_{ij} - p\alpha_{ij})(s_{ij} - p\alpha_{ij})} - \sqrt{\frac{2}{3}}kp = 0 \tag{3.12}$$

Relevant derivatives for this function are:

$$\frac{\partial f}{\partial \sigma_{ij}} = \frac{(s_{ij} - p\alpha_{ij}) + \frac{1}{3}\left[\alpha_{mn}(s_{mn} - p\alpha_{mn})\right]\delta_{ij}}{\sqrt{(s_{pq} - p\alpha_{pq})(s_{pq} - p\alpha_{pq})} + \sqrt{\frac{2}{27}}k\delta_{ij}} \tag{3.13}$$

$$\frac{\partial f}{\partial \alpha_{ij}} = \frac{-p(s_{ij} - p\alpha_{ij})}{\sqrt{(s_{pq} - p\alpha_{pq})(s_{pq} - p\alpha_{pq})}} \tag{3.14}$$

$$\frac{\partial f}{\partial k} = -\sqrt{\frac{2}{3}}kp \tag{3.15}$$

The plastic-flow rules derived from these models come from using the same functions as plastic-potential functions. Therefore, the plastic flow directions are given by $m_{ij} = \partial f / \partial \sigma_{ij}$. Note that in the Von-Mises model this results in a purely deviatoric plastic flow direction, while the Drucker-Prage model results in volumetric components which make the resulting material dilative in response.

3.1.6.2. *Hardening Laws.* The two types of variables used for this discussion are the backstress tensor α_{ij} and the scalar strength parameter k.

For the scalar strength parameter k the following linear hardening function is implemented:

$$h^k(\boldsymbol{m}) = H_k m_{eq} \qquad m_{eq} = \sqrt{\frac{2}{3}m_{ij}m_{ij}} \tag{3.16}$$

Where H_k is a hardening parameter and m_{eq} is called the 'equivalent' plastic-flow direction.

For the backstress, two hardening laws are presented. The first is a linear hardening law akin to the one presented:

$$h^\alpha_{ij}(\boldsymbol{m}) = H_\alpha m^{dev}_{ij} \qquad m^{dev}_{ij} = m_{ij} - \frac{1}{3}m_{kk}\delta ij \tag{3.17}$$

Where m^{dev}_{ij} is the deviatoric component of the plastic-flow direction. The second law is a non-linear hardening law known as the Armstrong-Frederick hardening rule. It is defined as:

$$h^\alpha_{ij}(\boldsymbol{m}) = \frac{2}{3}h_a m^{dev}_{ij} - c_r\sqrt{\frac{2}{3}m^{dev}_{kl}m^{dev}_{kl}}\alpha_{ij} \tag{3.18}$$

Here, h_a and c_r are model parameters.

For linear hardening, the backstress tensor can grow unboundedly resulting in a bi-linear type deviatoric response for the material. On the other hand, the Armstrong-Frederick rule results in a

saturation type evolution of the backstress. The growth of the backstress for Armstrong-Frederick is asymptotical approaching the limit:

$$\|\alpha_{ij}^{lim}\| = \sqrt{\alpha_{ij}^{lim}\alpha_{ij}^{lim}} = \sqrt{\frac{2}{3}}\frac{h_a}{c_r}. \tag{3.19}$$

For $c_r = 0$, the model degenerates into linear hardening. The yield function size, selected flow rule along with the values of c_r and h_r will prescribe the ultimate strength of a model featuring Armstrong-Frederick (Armstrong et al. (1966); Lubarda and Benson (2002); Jirasek and Bazant (2001)) hardening.

3.1.6.3. *Elasticity Laws.* Hookes law will be used for all present models. This is given by

$$E_{ijkl}(\boldsymbol{\sigma}) = \lambda\delta_{ij}\delta_{kl} + 2\mu\delta_{ik}\delta_{jl} \tag{3.20}$$

Where the Lamé parameters λ and μ are given in terms of the modulus of elasticity E and Poisson's ratio ν as:

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \qquad \mu = \frac{E}{2(1+\nu)} \tag{3.21}$$

A simple extension of this model is to make the elasticity modulus depend on the mean pressure:

$$E(p) = E_{ref}\left(\frac{p}{p_{ref}}\right)^a \tag{3.22}$$

Where E_{ref} is a reference elasticity modulus obtained at a reference pressure p_{ref} and the exponent a is a parameter controlling the growth of E with confinement. Typically $a \approx 0.5$ is used. E_{ref} can be calibrated using the low-strain shear modulus or shear-wave propagation estimations and a selected Poisson's ration. Models with this feature are beyond the scope of this work.

3.1.6.4. *Integration of Constitutive Equations.* Classical elasto-plasticity results in a rank six system of non-linear ordinary differential equations. In this work, an explicit first-order Euler integration algorithm with stress correction is implemented after Crisfield (1991). Its performance is demonstrated in the next section.

3.1.7. Examples of usage of the material classes. Using the presented framework, three constitutive models are instantiated.

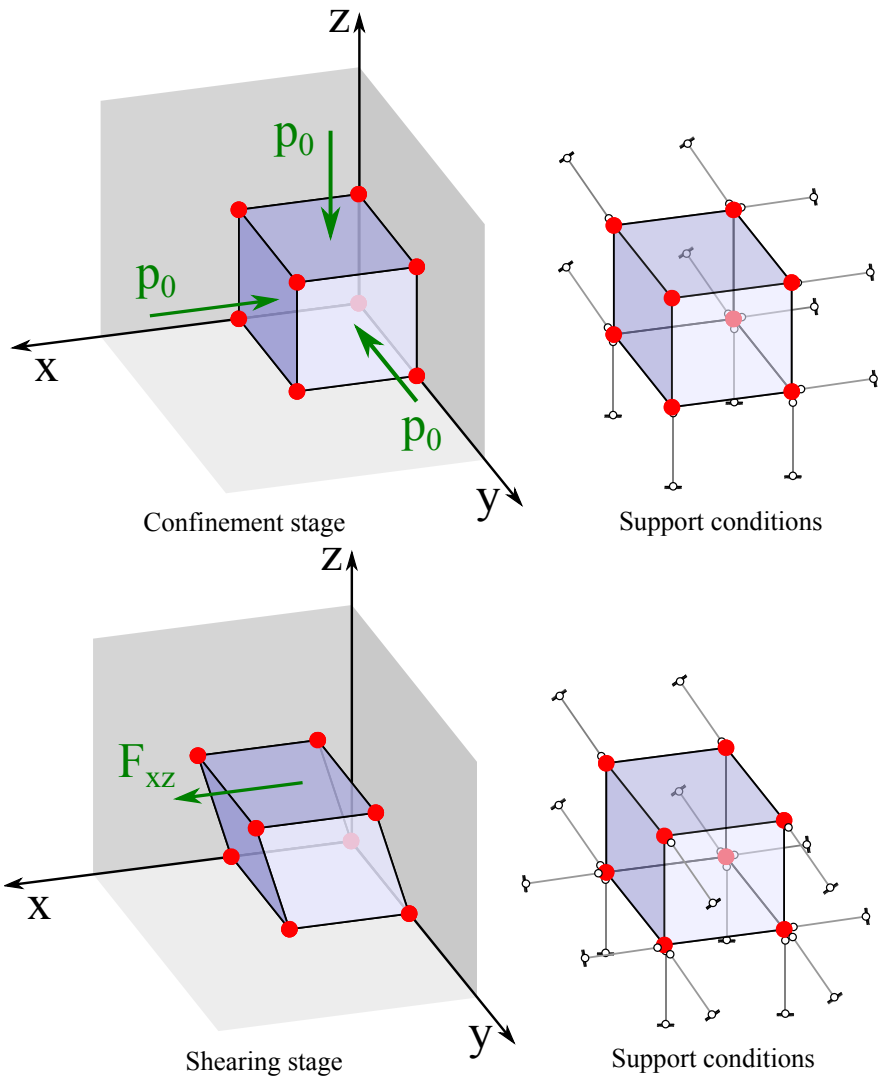


FIGURE 3.2. Configuration of the single element test for verification of the elasto-plasticity models developed within the framework.

- **Model 1** A Von-Mises model with associated plastic flow and linear kinematic and isotropic hardening.
- **Model 2** A Drucker-Prager model with associated plastic flow and linear kinematic and isotropic hardening.
- **Model 3** A Drucker-Prager model with non-associated plastic flow and linear isotropic hardening and Armstrong-Frederick kinematic hardening.

The first two models are very simple, and are useful mainly for verification of the implemented integration algorithms and instruction. The third model incorporates a more realistic hardening law, but closed-form solutions to its integration cannot be obtained, making its verification a harder task.

The elasto-plasticity models are incorporated within the object-oriented framework of the Real-ESSI Simulator program and subjected to single element test. The material properties are chosen as follows.

For all material models, E is chosen such that it represents a material with shear-wave propagation speed $V_s = 500\,[\mathrm{m/s}]$ with a density $\rho = 2000\,\mathrm{kg/m^3}$. Poisson's ratio is determined such that the compressional wave velocity is $V_p = 2 * V_s$, giving $\nu = 0.\bar{3}$. All models are confined to an isotropic state of stress with $p_0 = 250\,[\mathrm{kPa}]$. The Von-Mises strength parameter is chosen such that it represents an internal friction angle at initial yield of $\phi_y = 20^\circ$ for the chosen confinement pressure. Hardening is chosen such that a mobilized equivalent friction angle of $\phi_u = 30^\circ$ for the maximum shear stress applied. The Drucker-Prager model is set to have a friction angle at yield of $\phi_y = 25^\circ$ while the dimensionless hardening parameter is $H = 10$. Finally, the Drucker-Prager with Armstrong-Frederick hardening (DPAF) material is given an initial yield friction angle of $\phi_y = 5^\circ$ and the parameters c_r and h_a are chosen such that the post-yield tangent stiffness is comparable to the pre-yield one, and the ultimate strength is given by an implied friction angle of $\phi_u = 30^\circ$. With this $h_a = 1000$ and $c_r = 973$, both dimensionless. In this test, a singe eight-node brick featuring eight Gauss integration points is used to simulate a stage of isotropic consolidation followed by simple shearing. Figure 3.2 shows the setup of the single element and its boundary conditions.

With reference to said figure, in the first stage, confinement is achieved by applying pressure over three of the six faces which share a common vertex, converting pressure into equivalent nodal forces, while keeping the other three faces from detaching from the surface. In-plane displacements are allowed.

Once the desired confinement is obtained, the top face is freed to move in the X direction. Shear stresses are applied on the top face by converting them too to nodal forces. The analysis proceeds in a force-controlled fashion with equilibrium iterations (Newton-Rhapson method).

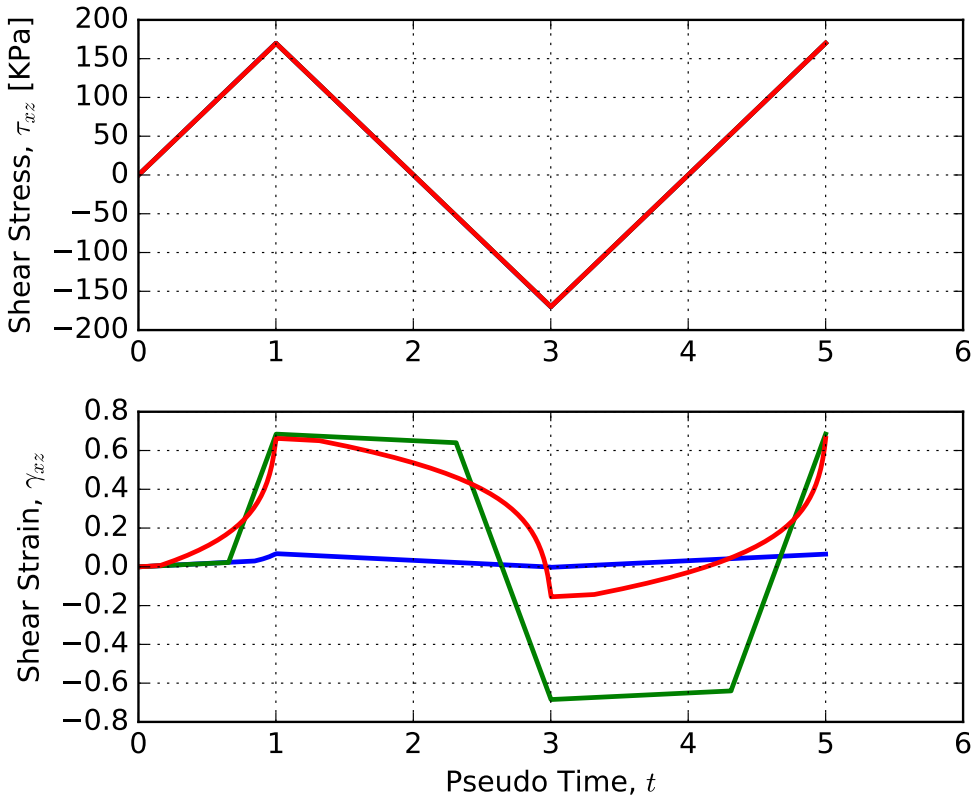


FIGURE 3.3. The stress pseudo-time history applied to the element after confinement to $p_0 = -250$ [kPa] and the resulting shear-strain pseudo-time history.

Several cycles of symmetric loading are applied to assess the cyclic behavior of the materials. Figure 3.3 shows the applied stress history and the obtained strain time-history for the element at one Gauss integration point (all Gauss-points respond in the same way).

Figure 3.4 shows plots of the shear-stress vs. shear strain response, the mean stress vs. volumetric-strain response, the deviatoric stress vs. mean stress response. Figure 3.5 shows the implied shear-modulus reduction curves for these responses. As expected, both Von-Mises and DPAF shear without volumetric changes, as can be seen in the absence of change in mean stress and volumetric strain. Both materials achieve the target strength and the mobilized ultimate friction angle. The Drucker-Prager material behaves with increasing mean stress due to the volumetric components of the associative plastic flow rule adopted. The material responds increasing mean stress instead of dilating because of the constrained shearing conditions which impede any

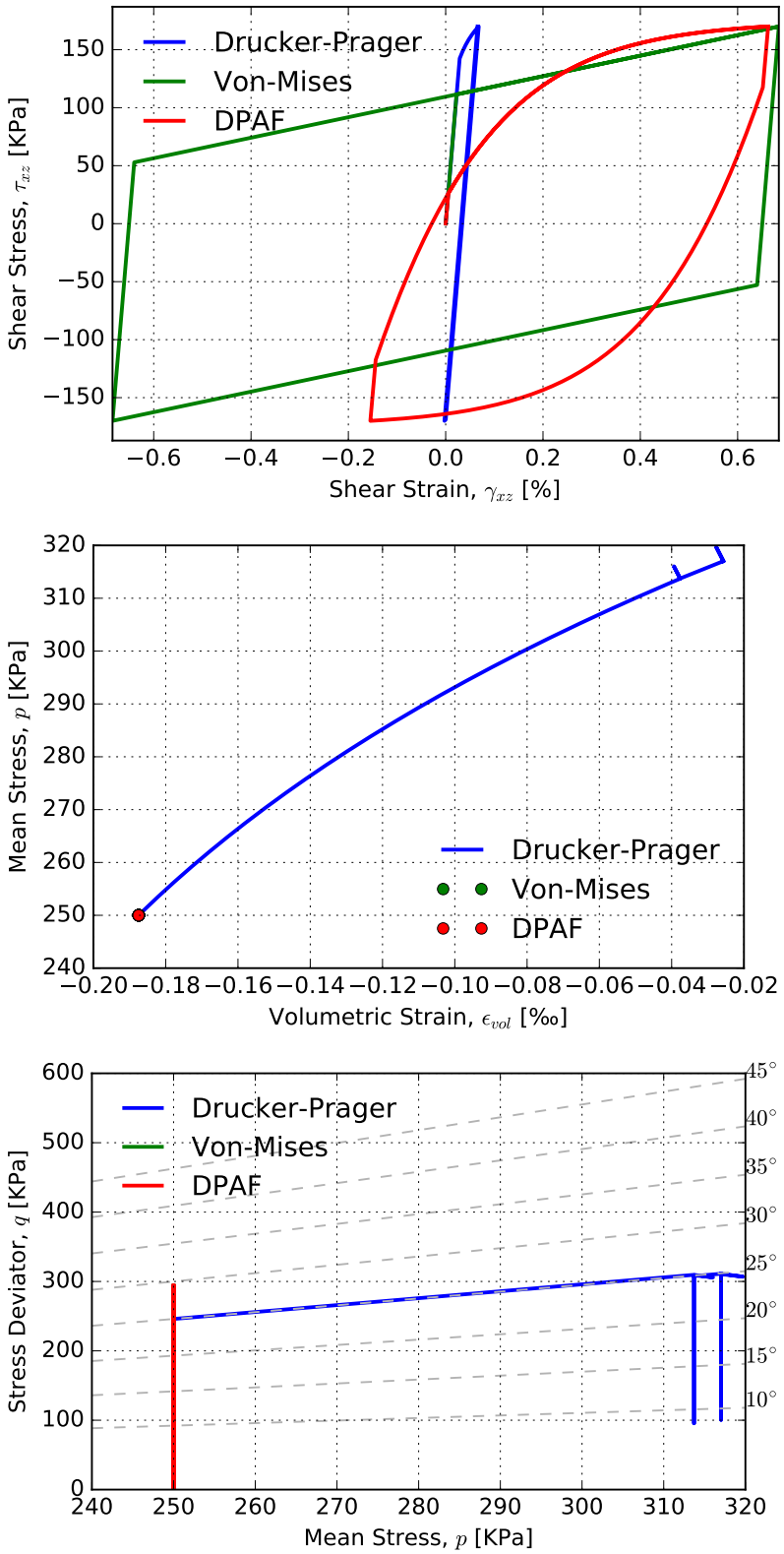


FIGURE 3.4. Stress-controlled response of the material models to a sawtooth (triangular) shaped shear-stress loading after confinement with $\sigma_{zz} = -250$ [kPa].

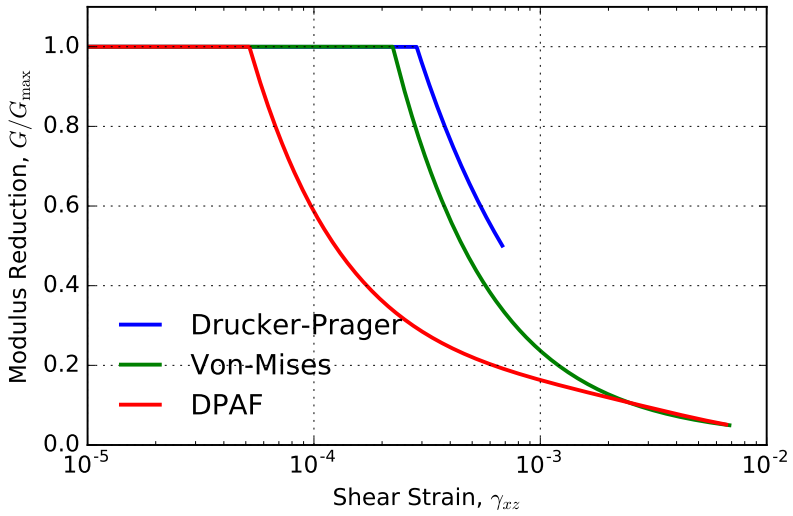


FIGURE 3.5. Implied modulus reduction curve G/G_{max} for the resulting material response.

volumetric expansion. As pressure increases and is maintained, the material no longer is able to reach the yield surface and ensuing cycles are linear albeit with a permanent deformation.

3.1.8. Conclusions. An efficient implementation of classical elasto-plasticity is presented which avoids usage of virtual function while providing a framework which encourages code modularity and reuse, hallmarks of maintainability. Advanced features of the C++11 standard are used to achieve this implementation. A demonstration of the expected performance benefits is presented with a reduced example that preserves the important features of the problem.

Using the framework, example elasto-plastic constitutive models are instanced and tested within the Real-ESSI simulation code via a single element test. Material response to a two-stage analysis, isotropic confinement followed by pure shearing, is as expected for the chosen material components.

3.2. Domain Specific Language for Finite Element Modeling and Simulation

3.2.1. Introduction. Creating a finite element analysis (FEA) model of a physical system is a very common task in civil engineering practice. At the very core, a finite element model approximates a system of partial differential equations (PDFs), called the equations of motion, which model the physics of the system. The mathematical description of a PDF requires the specification of the domain shape (geometry), the boundary conditions, constitutive laws (which relate state variables to the continuity equations expressed by the PDFs), and an initial condition or state. Additionally, users of FEA need to specify the methods and parameters that control the numerical solution of the problem at hand. Furthermore, many problems require the iterative evaluation of the solution when changing some of the governing parameters, such as shape, material properties, loading conditions, etc.

The two major methods used for model description are (i) text-based or scripting, and (ii) graphical user interface (GUI) based. GUIs provided an intuitive way of defining a model in which a rendering of the model and its properties is displayed and all properties are set by pointing and clicking within menus and other widgets. Scripting, on the other hand, uses a text file as an input to the FEA software. Specifications of scripts for FEA range from fixed format files to implementations of dedicated programming languages able to accomplish complex tasks. Table 3.3 shows a list of well known FEA modeling programs and their approach to modeling.

GUI based modeling is intuitive, because it appeals to human perception of the model. Models are built by drawing elements on grids, dragging and dropping properties, accessing menus to modify parameters, etc. It is limited, though, by the effort that the developer puts in making the features available to the user. As the core analysis code progresses, so must the GUI to reflect the changes, resulting in that GUI's lag behind in terms of features available. Also, due to the dependence of graphics, GUIs are usually not meant for remote operation, that is, modeling on a computer which is in another part of a network for example.

Scripting was the form of input available for the first FEA programs. It appeals less to human intuition while requiring that the user learns a programming language or a file format along with all the options in order to model. On the other hand it can provide more flexibility because it permits more complex simulation flow (through looping and conditionals), allows for parametric

Program	Modeling approach
Abaqus	Provides a GUI and uses the Python language for scripting
ADINA	GUI
ALGOR	GUI
FEAP	Provides its own DSL
FLAC	GUI and a scripting language
FreeFEM++	DSL based on mathemathical theory of FEM
COMSOL	GUI and interface into MATLAB
Code_Aster	GUI, and a scripting language based on Python
LS-DYNA	Fixed format file, GUI preprocessors are available
Nastran	Fixed format file, GUI preprocessors are available
OpenSEES	Extends the TCL language
Plaxis	GUI and also provides a command line
SAP2000/ETABS	GUI and provides open API in Visual Basic

TABLE 3.3. A non-exhaustive list of some FEA programs and their modeling paradigms.

problem descriptions (with user-defined variables), provides modularity in problem definition as well as model reuse. These features are achievable in GUIs, but usually result in cumbersome interfaces. A drawback of scripting based modeling is that it is very hard and error prone to input detailed models from a scripting language without additional aid.

A mixed, divide-and-conquer, approach has emerged and proved to be quite successful: the pre-processing/analysis/post-processing cycle. In this approach a pre-processor (or mesh-generator), usually GUI based, is used to define the model geometry and to create model files in the scripting language of another program (the analysis program), the modeler then uses these created files to create an analysis model and run the simulation. The output of the program is then fed into a post-processor, which sometimes can be the same pre-processor program. The advantage of this approach is that it separates the problem into three stages and at each stage a

specialized program is used. This allows building lean tools which perform a limited number of tasks, but perform them well, as opposed to having one application do everything. This is much akin to the philosophy which drove Unix operating system into success.

In this article we present the Finite Element Interpreter (F.E.I., styled FEI), a domain specific language for definition of finite element models, which serves as a scripting and interactive front-end to the FEA program RealESSI (Earthquake-Soil-Structure-Interaction) Simulator Program developed at the University of California, Davis. Details on the design goals and implementation are provided and discussed.

3.2.2. Design of the Domain Specific Language. A domain-specific language (DSL) is a small computer programming language which focuses on a particular problem domain. The main idea of domain specific language is to have a simple computer language in order to target a specific kind of problem rather than a general purpose language (Spinellis (2001)). Ideally, DSLs offer a simplified control experience compared to general-purpose programming languages, affording them an advantage in their particular domain of application in terms of user productivity, expressibility and maintenance of the tasks performed by the DSL.

Developing of DSLs is hard since it requires knowledge of both domain of application and also programming language development. In most cases developing a DSL for a particular application is either not considered at all or is postponed indefinitely. Although there are number of articles written on the development of DSLs, there are few number of articles on different methodologies of developing DSL and how to create them (Mernik et al. (2005)).

3.2.2.1. *Design goals.* The FEI scripting language was designed with five basic goals believed to greatly enhance finite element modeling in scripting form. These are:

(i) To encourage creation of self-documenting FE analysis scripts.
(ii) To promote understanding of non-linear finite element modeling.
(iii) Language should be aware of physical units.
(iv) Language syntax, features, and constants should simplify modeling while allowing for expressiveness.
(v) Be safe to execute in information-sensitive environments.

Within the context of this article, an analysis script is said to be self-documenting if it requires little or no addition of comments by the programmer to explain what is going on. Ideally, it will require little review of the documentation by the user if he/she is knowledgeable enough in FEA. Goal (i), therefore, is implemented in FEA by having the commands have the structure of the natural language used to explain the underlying theory. For example, the following shows the command to add a node.

```
add node # 1 at (1*m,-10*mm,3*yd)  with 3 dofs;
```

The command is self-explanatory in what it does and in what the positional arguments refer to, uses words which are commonly used in literature to refer to the components, parameters, and methods. It could be argued that this philosophy will lead to long commands which require much typing. The next command shows one such long command which is used to add a well known material model to the domain.

```
add material # 1 type camclay
    mass_density =  2300*kg/m^3
    reference_void_ratio = 0.21
    critical_stress_ratio_M = 1.2
    lambda =  1.0
    kappa = 0.2
    poisson_ratio =  0.3
    minimum_bulk_modulus =  1*kPa
    initial_mean_pressure =  0.1*Pa
    algorithm =  explicit
    number_of_subincrements = 10
    maximum_number_of_iterations =  1000
    tolerance_1 =  1e-4
```

Manually typing a command such as that is long and error prone. The proposed solution to this is not to type the command at all! Modern text editors are capable of sophisticated syntax highlighting and code completion, and many follow similar approaches on defining a new language. Given that storing and parsing long text files is not a big problem to today's computers, it is only obvious that the design of a new language take the performance of text editing software into account in order to make the language easier to interpret by a human as well as easier to maintain.

Another big idea behind the design of the language is that there should be no default values and that there should be no hidden features. Finite element users should know the models and methods involved to the extent that they should be able to provide reasonable values for all parameters. Many GUI- based, even scripting based, FE analysis programs provide typical default values for many parameters. It is our belief that providing default parameters is a dangerous practise in that it encourages the bad habit of not checking parameters, the modeller might rely on the judgment of whomever chose the default without thinking about the validity of it.

Goal (iv) is to provide physical unit safety or, at the very least, unit awareness. FE analysis is inherently unitless. In developing FE theory there is no regard whatosoever to the physical units implied in the computations. This ends up being one of a FEA modeller's Achille's heels, as it is a very common source of human error. Our idea is to implement unit safety directly into the DSL. In the FEI language, the basic type is floating-point number (i.e. a `double`) along with a physical unit, withall possible operations regarding these variables defined. The FEI API, like any FEA code, is unitless, so conversion to basic SI units (meter, kilogram, second) is done when arguments are passed to the API calls and analysis is done in that unit system. Feedback from early users of the FEI language (some practitioners and students) has been very positive towards this feature, as it has given them further insight into the physics of FE analysis. A contrived example of this feature is showcased in the previous '`add_node`' command where the coordinates are given in different units.

Goal (v) is to provide coding constructs as well as language variables (automatically set variables) which are useful and meaningful to FE modeling. A scripting language user expects the language to provide conveniences such as variables, mathematical operations and functions, conditionals (`if/else` statement), looping (`while` and `for`), sourcing of code (i.e. `include` statement or similar which reads code from another file and append them to the current code), and meaningful variables which simplify syntax (like a auto-set variable which holds the number of nodes in the model). For example, once nodes and elements have been set up in a model, it is typical to loop over the nodes or a subset of them to assign loads. This is a good place where some syntactic constructs can help both facilitate the work of the modeler as well as make his intent clear for others. More details on the considered language features in the next section.

Finally, for goal (vi), code safety is achieved by limiting what the language can do. The language does not feature a way to make system calls, like the `system` command present in many languages, or a way to execute arbitrary code. There are no input/output operations available to read and write from files, output from the FE code should be done through the code API and the provided bindings to the language. There is a basic implementation of character strings which are only needed for naming models and stages, and no string manipulation operations are provided. FEDS is not a general purpose programming language it is a domain specific one, so its features are only the ones relevant to FE modeling.

As a final remark, efficiency of the parser is not very significant for a FE program as it is a small overhead when compared to the actual computations done. Thus we have ample freedom to create a good parser.

3.2.2.2. *Language features.* The basic (and only) type in the language is a floating point number provided with a unit, which we call a 'quantity'. The language implements all possible operations between quantities, and also storing quantities in variables (and vectors of quantities). The following listing shows the use of units to compute the pressure exerted on a circular area of diameter 10 cm by a force of 10 N.

```
1 Force = 10*N;
2 diameter = 10*cm;
3 print Area = pi*(diameter/2)^2;
4 print pres = Force/Area;
5 print pres in kPa;
```

Command line output

```
0.00785398 [m^2]
1273.24 [kg*m^-1*s^-2]
1.27324 [kPa]
```

Basic units are stored as 'locked variables' is FEDS , that is, variables that have been set in advance and cannot be modified. Additional units and quantities can be defined in terms of these basic variables. In the example, the force `F` has been defined in terms of the predefined variable `N`, which is the Newton. Appendix B.1 shows a list of predefined basic variables.

Also shown in the preceding example is the `print` command which shows the contents of a variable. Please note that the units of the variables are shown between the brackets '`[]`', and if the expression has no units and empty set of brackets is shown. Values are shown by default in standard SI units (since they are all expressed in that system), but line 5 shows how to use the `print` command to show the variable in other units using the '`in`' command. The `print`

statement can also print strings (defined with double- quotes) and several expressions in a line separating them by comma.

Standard arithmetical operations (`+,-,*,/`) are allowed for all variables. The remainder (or modulus) binary operation, `a % b = a mod b`, is defined. Also, the exponentiation operation is available thorugh the "`^`" operator, with the caveat that the exponent must be a unitless integer. Assignment, is done using the '`=`' operator. C++ style compound operators also avaliable are: `+=, -=,*=, /=, %=`.

Special mathematical functions (`cos`, `sin`, `tan`, `cosh`, `sinh`, `tanh`, `acos`, `asin`, `atan`, `atan2`, `sqrt`, `exp`, `log`, `log10`, `ceil`, `fabs`, `floor`) are available for unitless inputs.

Name	Symbol	Name	Symbol
Greater than	>	Less than	<
Not equal to	<>	Equal to	==
Greater than or equal to	>=	Less than or equal to	<=

TABLE 3.4. Logical operations defined in 卍 .

Available logical operations are summarized above. All logical operations return an adimensional 1 if the comparison is true or a 0 if not. Compared quantities must have the same units, else comparison returns 0. For compound logical statements one can use the 'logical and' (`&`) or the logical or' (`|`) operators, which only work on adimensional quantities. Available, also, are logical tests to check for certain unit types. For example `isLength(var)` checks whether the variable `var` has units of length and returns a unitless 1 if true or a zero if false, regardless of the actual value. Available tests are: `isAdimensional()`, `isMass()`, `isLength()`, `isTime()`, `isArea()`, `isVolume()`, `isForce()`, `isEnergy()`, `isTorque()`, `isPressure()`, `isBodyForce()`, and `isDensity()`.

Conditinal branching is possible through `if/else` statements which are exemplified in the next code listing.

```
a = 10*m;
if( isForce(a) )
{
  print "a is a force quantity";
  print a;
}
else
{
  print "a is not a force quantity";
  print a;
}

if (a > (5*m))
{
  print "a is greater than 5m";
}
else
{
  print "a is less than or equal to 5m";
}
```

Command line output

```
a is not a force quantity
a = 10 [m]
a is greater than 5m
```

Looping is also possible with the `while` statement exemplified in the next fragment. Also available are `for` loops, but these have a non-traditional meaning in FEI which is to loop over system components or a subset of them, more will be explained on the following sections.

```
i = 1;

while(i <= 5)
{
    print i;
    i += 1;
}
```

Command line output

```
i = 1 []
i = 2 []
i = 3 []
i = 4 []
i = 5 []
```

Note that the `if`, `while`, and `for` statements require the use of braces to enlose the block of code.

The `include` statement allows incorporating another script in the line where the statement is issued. The syntax is: `include "path/to/include_script.fei";`. This is a very necesary feature, because FE models can grow vastly in number of nodes and elements (which might be generated externally by a meshing program) and it is useful to place these in other files, and keep the 'main' script as lean as possible for clarity.

Line code comments are achieved by using the double slash (‘//’). These can be located anywhere, and the rest of the line will be commented out.

Finally, the `bye;` command is issued at the end of all analysis, which makes sure to free all memory and flush all buffered output data before exiting the program. In case needed there is also a `restart;` command that does the same memory erasing operations but allows a new analysis to continue. During a restart, the command language variables are not erased, although those language variables and constants (like number of nodes) are affected.

3.2.2.3. *Modeling features.* In what follows triangle brackets <> encountered within code listings denote a placeholder for a variable or expression to be input into a command. The contents of the brackets define the units that that variable needs to have. For example `<l^2>` means that the variable needs to have units of length squared. All units are expressed as a combination of units of length (`l`), mass (`m`), and time (`t`). A force, therefore, will have a escription of type `<m * l * t^-2>`. Sometimes the units of the variable will depend on the type of command being placed, in that case the empty triangular brackets are used <> An adimensional quantity is expressed as `<.>`. Also, the rectangular brackets `[]` are used to denote a place in the code where a variable language keyword may be placed, and a hint to the type of keyword is placed inside the brackets. B.2 in the appendix contains a list of keywords available for each bracket.

Every model has to have a name in the FEA language, so every analysis begins with a model name declaration by issuing the command: `new model "name_string";` which will give a global title to the analysis to be performed.

A model is composed of nodes, elements that connect the nodes, materials which make up each element, boundary conditions and constraints, and forces (point, body, and surface loads). Additionaly, one needs to specify the solution strategy, the method for solving a linear system of equations, a method to check for convergence, and the number of simulation steps to produe. The following subsections show how the language facilitates each of these steps, but assumes that the user is knowledgeable in FEA.

Node definition

To define a node, the command pattern is shown below

```
1   add node # <.> at (<l>, <l>, <l>) with <.> dofs;
```

Nodes can have different number of degrees-of-freedom (dofs), but care must be taken to be using the correct number of dofs for the elements which might connect to the node.

Material definition

Materials must be defined before creating the elements. The material definition commands follow the pattern shown below

```
1  add material # <.> type [material_type] [parameter_1] = <> [↩
       parameter_2] = <> ... [parameter_N] = <>;
```

After the `material_type` name follows a list of parameters which is different for each material. An example for the 'Cam-Clay' constitutive model was shown in page 50.

Element definition Elements are defined with the pattern shown below.

```
1  add element # <.> type [element_type] with nodes (<.>, <.>, ... ↩
       , <.>) use material # <.>;
```

Some elements, like the two-phase porous u–p–U elements, have a parameter list. These are appended at the end of the command similar to what is done with materials.

Constraints

The most basic constraint is the homogeneous holonomic constrain that sets a single degree of freedom to be zero. This is achieved with the command

```
1  fix node # <.> dofs [dof];
```

FEI language provides keywords such as `ux` to specify degrees of freedom, in this case it refers to the displacement in the x direction. Also available are the y and z displacement DOFs (`uy` and `uz`), the rotational degrees of freedom about all axis (`rx`, `ry`, and `rz`), and the displacements of the fluid-phase for u–p–U elements (`Ux`, `Uy`, and `Uz`) along with the pressure DOF (`p`).

Loading

The four commands shown below are a subset of the available options for loading in FEI . The first one applies a single, linearly increasing, load at a given node. The second one takes input from a filename, and is appropriate when loading histories other than linear are required. The third and fourth cases go together to produce self weight loading.

```
1 add load # <.> to node # <.> type linear [force_keyword] = <>;
2 add load # <> to node # <> type path_series [force_keyword] =  <> ↩
    series_file = "path/to/time_series_filename";
3 add acceleration field #  ax = <1*t^-2>  ay = <1*t^-2> az = <1*t↩
    ^-2> ;
4 add load # <.> to element # <.> type self_weight use acceleration ↩
    field # <.>;
```

Commands exist to produce loading from surface loads, displacement time-histories, and loadings consistent with the Domain Reduction Method (DRM) for realistic earthquake response analysis.

Querying elements and nodes for data to use in analysis

Sometimes it is needed to modify the conditions of analysis depending on the progress and results obtained during the analysis, which are not known in advance. To do this, FEI Language provides a convenient syntax to access information about domain components, in a read-only manner.

```
1  node[<.>].[node_data_keyword];
2  element[<.>].[element_data_keyword];
```

For example, on line 1 above, `node_keyword` can take the values `ux`, `uy`, etc. to access the current value of a certain DOF. Other keywords are available to check the nodal unbalance, nodal coordinates and other relevant information. Shown in line 2 is the syntax for elements. For elements one can access stresses and strains at different Gauss points, connectivity, coordinates of nodes, tangent stiffness matrix components, among other options.

Additionally, FEI language provides automatically set variables for some important domain parameters. As an example, the number of declared nodes is contained in `NumberOfNodes`↩, the number of elements in `NumberOfNodes`, and the current simulation time is contained in `ModelTime`.

Looping over elements and nodes

FEI provides shorthand syntax to loop over all nodes and elements, which is often necesary to apply loads or other boundary conditions. Furthermore, within the ESSI simulator, nodes and

elements can be assigned group lables to distinguish certain areas of the model; the looping can be done for a specific subset of elements or nodes belonging a certain group.

Below, the syntax for looping over nodes and printing the nodal coordinates is exemplified.

```
for n in nodes
{
  print "Node number ", n;
  print node[n].x;
  print node[n].y;
  print node[n].z;
}
```

Command line output

```
Node number 1
 0.00 [m]
 0.00 [m]
 0.00 [m]
Node number 2
 0.1 [m]
-0.2 [m]
 5.7 [m]
Node number 3
 3.4 [m]
 0.2 [m]
-1.7 [m]
...
```

Simulation options

Control of simulation conditions such as tolerances, time steps, algorithms, integrators and such is achieved through simulation option setting commands such as those shown in below.

```
define dynamic integrator [dynamic_integrator_keyword] with [↩
   parameter_1] = <> ... ;
define static integrator [static_integrator_keyword] with [↩
   parameter_1] = <> ... ;
define convergence test [convergence_test_keyword] with [↩
   parameter_1] = <> ... ;
define algorithm [algorithm_keyword];
define solver [solver_keyword];
```

Running the analysis

Finally, the execution of the analysis is done through `simulate` family of commands, two examples of which are given in the following box.

```
simulate <.> steps using static algorithm;
simulate <.> steps using transient algorithm time_step = <time>;
```

3.3. Parallel Finite Element Method

Appendix A presents an introduction to the finite element method without regards to how it is to be implemented. In this section some techniques for parallelization which have the highest impact on performance of the method are discussed. As shall be seen, nonlinear FEM offers plenty of opportunities for parallelization.

3.3.1. Mesh partitioning. In non-linear FEM there is a substantial amount of time spent in computing the material response. That is, for a given increment in displacements which produce an increment in strains at each element Gauss point, a considerable amount of time is spent computing the stress increment and the corresponding tangent elastic modulus tensor. This provides a source of parallelism because all these computations are independent of each other, or as commonly termed in the high-performance-computing field 'embarrassingly parallel'[5].

To exploit this parallelism it is necessary to distribute the, independent, stress increment computations across several processes. The logical way to do this is to partition the finite element mesh in as many partitions as processes are available. Now, in doing this, it is important to take into account the connectivity between elements so that the mesh can be partitioned in a way that minimizes cuts to the mesh. Cuts signify, as will be seen in the next subsection, inter-process communication which is slow and must be minimized when devising a partitioning.

A graph representing element connectivity is built to aid with the partitioning using the `METIS` and `ParMETIS` libraries (Karypis and Kumar, 1995). Connectivity between neighboring elements is determined based on the number of nodes of the elements. If the elements are of the same type, they will be considered connected if they share a face. Figure 3.6 illustrates the connectivity criterion for 8-node brick (hexahedron) elements. For elements of different type, connectivity will be determined using the criterion of face-sharing for the element with the lowest number of nodes. For example, a beam will be connected to a brick always as long as one of its nodes is common with a brick, a twenty-seven node brick will be connected with an eight-node brick if the eight node brick has one of its faces (four nodes) adjacent to part of the face of the twenty-seven node brick, and so on.

[5] And the only thing embarrassing about the computations is if parallelism is not exploited.

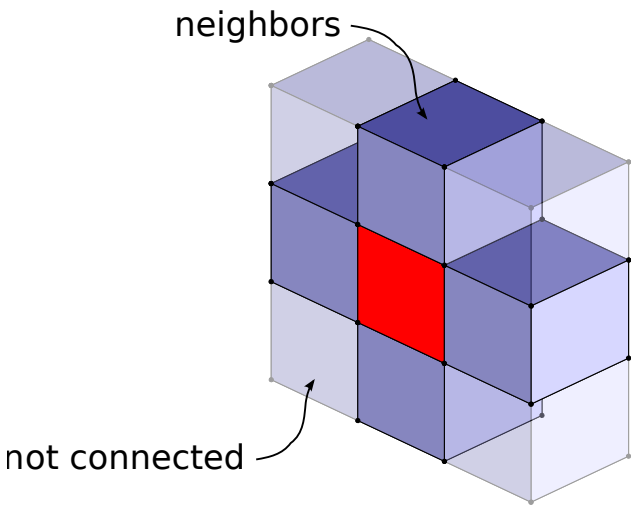


FIGURE 3.6. Connectivity criterion for 8 node bricks.

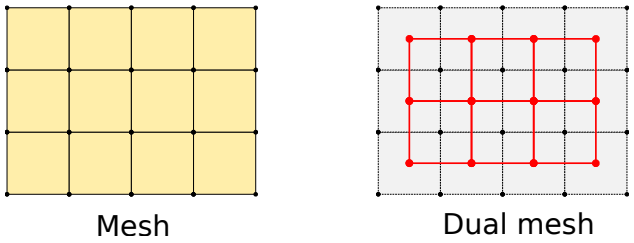


FIGURE 3.7. Finite element mesh and its dual graph

The graph constructed this way, called the 'dual' graph of the finite element mesh, has the elements as its nodes while edges represent element connectivity. Figure 3.7 shows a representation of the concept of the dual graph of a mesh. The partition of this graph yields a partition of the finite element mesh.

For a mesh with several element types and material models, different elements will take different times in completing their updating. Therefore, as shown by Jeremić and Jie (2007), it is beneficial to weight the graph nodes (elements) by the time taken to update and partition such that the summation of time on the partitions approximately balances.

This is the main concept behind the plastic domain decomposition which provides a perfect balancing scheme for non-linear plasticity computations. Usually this partition is not stable in time, meaning that as plasticity occurs at different parts of the model at different time-steps, the optimal partition will change. The cost of re-partitioning, which involves sending elements to

neighboring processes possibly through the network, must be taken into account when deciding whether to repartition. Additionally, the cost of moving an element will change depending on where the target process is being physically run. If it is within the same computational node the cost will be lower than if network communication must be used to transmit the element between computational nodes. Additional architectural considerations might be needed to determine the cost of transfer within the same computational node even[6].

Repartitioning makes sense if the time taken to do updates are significantly variable[7] during the life of the simulation. If the time taken by the elements to update their constitutive law is constant, there is no need for re-partitioning and the initial partition (still weighted with the time taken to update different elements) will provide a balanced parallelization throughout the simulation.

Figure 3.8 shows a typical partition of a 2-D mesh of eight-node bricks into 25 partitions. The partitioning process is an NP-complete problem and is approached using heuristics. Therefore, the partitioned graph is only approximately balanced for real applications. This proves to be enough for this type of computations.

Along with the distribution of elements, also nodes will be distributed. This implies that some nodes, those located along partition boundaries, will be duplicated because they need to exist across multiple processors.

3.3.2. Stiffness assembly and distributed solution of system of equations. Once the state of the material points is advanced the tangent stiffness matrix has to be assembled from the tangent elastic moduli tensor of each element. With the material points distributed there is also a partition of the system of equations because nodes (and the degrees-of-freedom defined at those nodes) will need to be distributed too. This results in the natural partitioning of the system of equations with the added complication that some degrees-of-freedom (DOFs) are duplicated

[6]For example, at the time of writing of this dissertation, the non-uniform memory access (NUMA) architecture is becoming common for high-end servers and HPC clusters. In NUMA architectures the memory address space is split between NUMA nodes, access of the address space within the same NUMA node is faster than access of memory belonging to a different NUMA node. Therefore, transferring elements between NUMA nodes might be slower than within the same node. This might prove important for extremely high performance computations.

[7]An important example of when this occurs is when material models implement stress integration with error control. Looping at the Gauss-point level in order to achieve integration tolerance will take a variable number of time steps during the simulation.

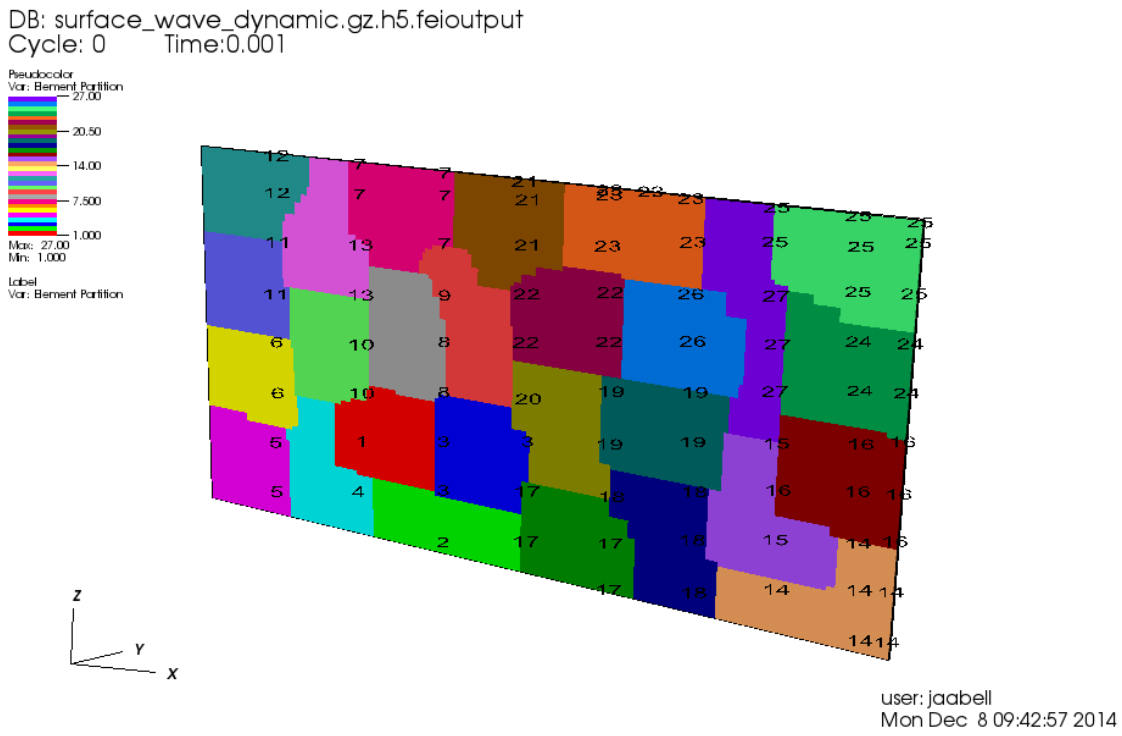


FIGURE 3.8. Visualization of a typical METIS partition of a 2-D mesh into 25 partitions.

across processes. A decision must be made as to which process will own and store the solution corresponding to each DOF.

Figure 3.9 shows the sparsity pattern of a typical system-of-equations from finite elements. There is no single solution to the system-of-equation partitioning problem. RealESSI Simulator uses PETSc libraries (Balay et al., 2015a, 1997, 2015b) to assemble, store and solve the linear system-of-equations.

Each processor stores a set of rows of the system of equations, a contiguous range of the solution vector and right-hand-side vector. Some columns of the local portion of the matrix belong to the same process, while the rest all belong to different processes. Therefore, assembly proceeds by each process writing their own components. Local components are added directly to the local representation of the matrix, while components belonging to the other processors are cached for later assembly on remote processes.

As shown in (Jeremić and Jie, 2007), a connectivity-based mesh partitioning enforces locality of data within partitions, which minimizes communication. The resulting sparsity pattern is

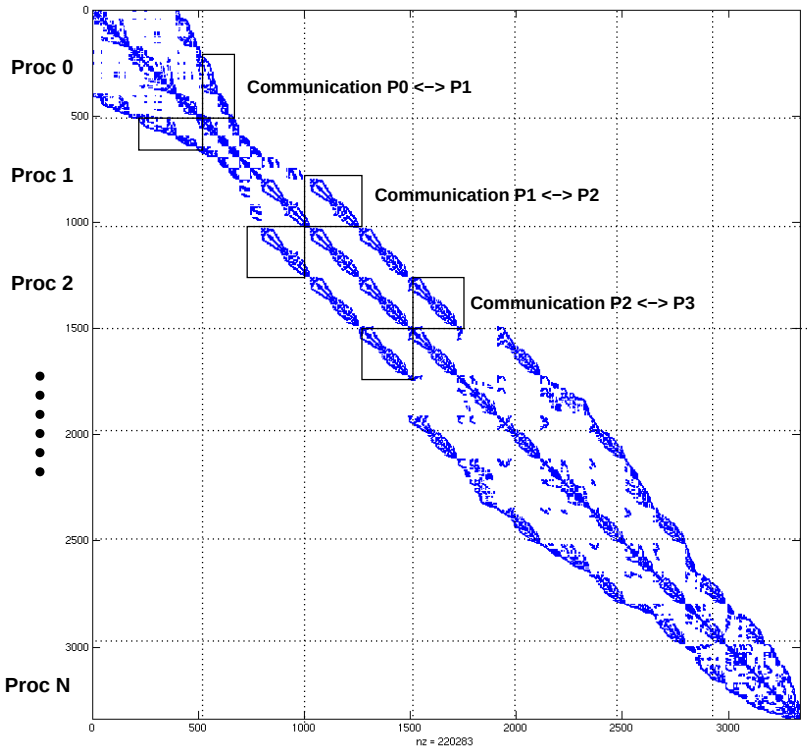


FIGURE 3.9. Typical stiffness matrix for a FEM mesh, showing how distributing rows amongst processors generates communication between them.

much like that displayed in Figure 3.9, except that communication will involve more than once process.

It is worth mentioning that load-balancing the system-of-equation will be achieved using the DOF connectivity graph (which pairs of DOFS are coupled through the system matrix), while balancing the mesh involves element connectivity and time taken at each element. Thus, balance of system-of-equations cannot be achieved simultaneously with balance of element updating, in other words if perfect mesh balance is achieved then probably the SOE will be out of balance. Arguably the greatest benefit comes from prioritizing the balancing of element update computations for two reasons. First, these computations are embarrassingly parallel which allows perfect speedup when executing in parallel, and second, because it is the slowest part of the computation it has the greatest impact in overall performance of the code.

3.3.3. Parallel Computation of DRM forces. Equation 2.7 is not the most efficient way to obtain the DRM forces. It is most efficient to *assemble* the DRM forces in an array much like the way the stiffness matrix is assembled. This is done in an element-by-element fashion as follows

$$\boldsymbol{P}_b^{\text{eff}} = \sum_{k_e=1}^{N_e} \boldsymbol{L}^{k_e} \left[-\boldsymbol{M}_{be}^{k_e} \ddot{\boldsymbol{u}}_{e0}^{k_e} - \boldsymbol{K}_{be}^{k_e} \boldsymbol{u}_{e0}^{k_e} \right] \tag{3.23}$$

$$\boldsymbol{P}_e^{\text{eff}} = \sum_{k_e=1}^{N_e} \boldsymbol{L}^{k_e} \left[\quad \boldsymbol{M}_{eb}^{k_e} \ddot{\boldsymbol{u}}_{b0}^{k_e} + \boldsymbol{K}_{eb}^{k_e} \boldsymbol{u}_{b0}^{k_e} \right] \tag{3.24}$$

Where k_e is just an index referring to each element, matrices $\boldsymbol{K}^{k_e}$ and $\boldsymbol{M}^{k_e}$ are the k_e-th element's stiffness and mass matrices respectively, and $\boldsymbol{L}^{k_e}$ is a transformation matrix which 'assembles' the local DRM forces into the global force vector. This matrix would not actually be built, just the DOF incidence graph is used to place the forces into the right entries of the global array.

In a parallel implementation the finite elements will be distributed across processes, each process would the have to carry out their own portion of this sum and assemble it into their own right-hand (forcing) side of the system-of-equations. A collective reduction operation (i.e. `MPI_AllReduce`) operation is then used to assemble the final right hand side vector for use with the solution of the system.

CHAPTER 4

Generation of Seismic Motions by Physical Modeling

4.1. Problem statement

As discussed in Chapter 2, application of the domain reduction method requires knowledge of seismic displacements, accelerations and possibly velocities in time for the nodes located on the boundary of the DRM layer. It is not practical to obtain this information by physical measurements, therefore, they must modeled mathematically and simulated numerically. What is needed, then, is a rational methodology by which to obtain said displacements starting from a specification of the earthquake source and crustal properties. In other words, proper of modeling the earthquake and propagation of seismic waves towards the site of interest is needed.

Elastodynamic theory has been used successfully to model the seismic wave-field produced by earthquakes within Earth's crust (Aki and Richards, 2002; Scholz, 2002). Given an adequate earthquake source description and a model of the crustal geology it is possible to compute, usually through a numerical evaluation of the elastodynamic equations, the response of material points within the crust to earthquake source models.

Under the assumption that Earth's crust is comprised of a continuous substance, e.g. it is an infinitely divisible material, and disregarding point rotations, the analysis of the requirements of dynamical equilibrium and conservation of angular momentum of an infinitesimal element yields the equation of motion for a general continuous solid, shown in Equation 4.1 in vector notation.

$$\rho \ddot{\mathbf{u}} = \boldsymbol{\nabla} \cdot \boldsymbol{\sigma} + \mathbf{F} \tag{4.1}$$

The equation above can also be written in terms of its Cartesian tensor components using the Einstein summation convention (Einstein, 1916), shown below, and which will be used throughout the rest of the section.

$$\rho \ddot{u}_i = \sigma_{ji,j} + F_i \tag{4.2}$$

Here $\boldsymbol{u}$ is the displacement vector of a material point, ρ is the mass density of the material, $\boldsymbol{\sigma}$ is the Cauchy stress tensor, and $\boldsymbol{F}$ collects all body forces acting on the system. All these defined symbols correspond to spatially varying fields. Additionally $\boldsymbol{u}$ and $\boldsymbol{\sigma}$ are functions of time. The body forces may be functions of time in the case of transient events or constant in time as in the case of gravitational force. Singular sources (also called point sources) are a special type of body force which are null everywhere except on a single point in the domain. In time, they might exhibit a finite duration or also be zero everywhere except on a time-point. Singular sources are very useful tool to characterize and describe the earthquake source, their treatment will be explained in greater detail in the next sub-section.

Equation 4.1 or 4.2 hold within a three-dimensional domain Ω which, for seismic applications at local or regional scales, is assumed to be unbounded. Seismic waves are, therefore, expected to travel out of the domain never to return. In reality Earth is bounded and waves do return eventually, albeit heavily attenuated .

For a localized earthquake source, e.g. one which does not extend indefinitely, the amplitude of the generated body wave-field decays with the inverse of the distance to the fault squared (Aki and Richards, 2002). Displacement magnitudes decay rather fast when compared to geological scales, even for near-fault applications. For this reason, with exception of the immediate vicinity of the rupturing fault, the small-strain assumption is taken to hold. The small-deformation strain tensor, or simply the 'strain tensor', is defined through the symmetrization of deformation gradient.

$$\epsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right) \tag{4.3}$$

To relate the state of strain, represented by ϵ_{ij} with the state of stress, represented by σ_{ij}, an assumption about the material constitutive behavior must be made. Linear elasticity, or Hooke's Law, has been shown to be adequate for a large number of seismic problems for two reasons: first, the small amplitude of seismic waves is not capable of mobilizing plastic behavior of crustal rock and, second, even at moderate depths, high confinement pressures of the soil column warrant high strengths for the materials involved, making non-linear behavior unlikely. This has been confirmed by the accuracy with which seismic wave predictions have been made based solely on Hooke's law. Equation 4.4 is the statement of Hooke's law which relates strains to stresses

linearly thorough a rank-four tensor of elastic constants E_{ijkl}.

$$\sigma_{ij} = E_{ijkl}\epsilon_{kl} \tag{4.4}$$

Further, the material is often assumed to be point-wise isotropic in which case the elasticity tensor can be fully specified using two moduli.

$$E_{ijkl} = \lambda\delta_{ij}\delta_{kl} + 2\mu\delta_{ik}\delta_{jl} \tag{4.5}$$

Where λ and μ are known as Lamé's parameters. μ is also known as the shear modulus of the material and alternatively denoted as G. This leaves only three parameters related to material to be defined for a complete description of the problem: ρ, λ and μ. These parameters are indirectly measured by observing the speed of different wave components as they propagate through the crust. In particular the speed of P-Waves V_p and that of S-waves V_s, along with the density are used to specify the other elastic parameters via:

$$\mu = \rho V_s^2 \qquad\qquad \lambda = M - 2\mu \tag{4.6}$$

where M is the P-wave modulus defined as

$$M = \rho V_p^2 \tag{4.7}$$

In addition to elasticity it has been observed that the crust shows a visco-elastic behavior due to which waves with distinct frequency content are attenuated with varying strength. In other words, wave attenuation has a frequency-dependent behavior along with the geometric-dependent behavior due to distance to the source. This frequency dependent behavior is modeled conceptually using the so-called 'quality factor', Q (Aki and Richards, 2002), which is an additional parameter to the wave solutions defined in the frequency domain

$$\boldsymbol{U}(\omega, \boldsymbol{x}) = \boldsymbol{U}_0(\omega, \boldsymbol{x}) \exp\left\{-\frac{\omega R(\boldsymbol{x})}{2Q}\right\} \tag{4.8}$$

Where $\boldsymbol{U}(\omega, \boldsymbol{x})$ is the Fourier transform of the response at a given point $\boldsymbol{x}$ away from the source, evaluated at angular frequency ω. $\boldsymbol{U_0}(\omega, \boldsymbol{x})$ is the response neglecting attenuation, $R(\boldsymbol{x})$ is the

distance from source to the point $\boldsymbol{x}$ at which response is being measured and Q is the quality factor. In reality there are Q factors associated with different phases of the waves. Hence there will be a different Q_S for S-waves and Q_P for P-waves and even surface waves will have different quality factors. It is important to note, however, that this model of anelastic attenuation is not physically realizable because its defined in the frequency domain where it was first observed and from where it is usually computed. Modifying the equation of motion to include attenuation effects of the type described by Q operators is possible only in an approximate sense (Petersson, 2012) and leads to a visco-elastic model. Nevertheless, anelastic attenuation is very important and calibration of Q factors must be done regionally for each application based on recorded motions.

Semblat (1997) provides a useful rheological interpretation of the quality factor Q in terms of Rayleigh viscous damping ξ such that for weak to moderate Rayleigh damping:

$$\frac{1}{Q} \approx 2\xi \tag{4.9}$$

To complete the mathematical model of elastodynamics the specification of the boundary conditions is needed. The first boundary condition is that of 'free surface' condition at the top of the domain. This translates to

$$\sigma_{ij} n_j = 0 \tag{4.10}$$

where n_i are the components of the unit normal vector to the surface. This equation is the same for the case of flat surface topography (in which case $n_1 = 0$, $n_2 = 0$ and $n_3 = 1$) as well as any general topography given by a height function $h = h(x, y)$.

It is further assumed, as discussed previously, that the domain is unbounded in the other directions. Given that a finite extension and duration source will produce a wave-field which decays with time and distance to source, it is assumed that all solutions will decay to zero as they approach infinity in any direction except towards the surface.

$$\lim_{r \to \infty} u_i = 0, \quad \forall \theta, \phi \tag{4.11}$$

Where r is distance from the source and θ and ϕ denote azimuth and elevation angles away from the source. In practise, depending on the method of simulation chosen to solve Equation 4.1, this condition will only be approximately satisfied.

Finally, the initial condition is assumed to be zero displacements and velocities for all points in the domain at the beginning of simulation time.

$$\boldsymbol{u}(\boldsymbol{x}, 0) = \dot{\boldsymbol{u}}(\boldsymbol{x}, 0) = 0 \tag{4.12}$$

Equations 4.2, 4.3, 4.4, 4.10, and 4.12 completely define the elastodynamics problem. Some analytical or semi-analytical solutions exist for this problem in some ideal scenarios such as the static case without layering subject to a point source (Okada, 1986) or the horizontally layered case (He et al., 2003; Zhu and Rivera, 2002). For a general setting with realistic crustal properties these equations must be solved numerically. Traditionally, this is done either by the finite element methods or finite differences.

The conditions for obtaining a numerical solution to the elastodynamic problem will be discussed next.

4.2. Conditions for correct solution of the elastodynamic wave equation

Solving the elastodynamic equations numerically, by any method, requires some form of discretization of the continuous mathematical model into grid points or cells, and the time dimension into time-steps. Alternatively, the problem might be posed in the frequency-space or frequency-wavenumber domains. In any case, each method will imply directly or indirectly a spatial and temporal discretization of the domain. The spacing of the grid points and chosen time-step size directly relate to the computational demands for obtaining the solution. Such parameters must be chosen thoughtfully to achieve a balance between precision requirements and computational capabilities of the infrastructure where the simulation will be performed as well as possible time constraints.

In three-dimensional space, for a rectangular domain of a given size with uniform discretization, the number of grid points will grow with the inverse cube of the grid spacing. This is the same rate at which memory requirements and run-time will grow. Furthermore, for some

storage schemes the amount of data generated might grow at an equally large rate. For a given time length of simulation, run-time grows linearly with the inverse of the time-step, as well as storage requirements.

The grid spacing in discretized models of wave propagation should be determined by the highest frequency which the modeler wishes to represent accurately. From discrete signal processing, the highest frequency representable is the Nyquist frequency (f_{nyq}) (Oppenheim et al., 1997), which is half of the sampling frequency ($1/2\mathrm{d}t$). This does not, however, correspond to the maximum *resolvable* frequency ($f_{\max}$), that is, the highest frequency at which the signal amplitude can be known with reasonable precision. That frequency will be related to the properties of the dynamical system modeled, in particular the waves featuring shortest wavelength encountered within the model must be correctly resolved, as well as the discretization scheme used.

The shortest wavelength will be associated with the slowest layer in the model, which are usually the topmost crustal layers. If the maximum frequency to be resolved is $f_{\max}$ and the slowest wave speed is β, then the wavelength of a wave at this frequency is $\lambda_{\min} = \beta / f_{\max}$. If $\mathrm{d}r$ is the largest spacing between two adjacent grid points, then adequate resolution of this wave requires

$$\mathrm{d}r \leq \frac{\lambda_{\min}}{N_{pts}} = \frac{\beta}{N_{pts} f_{\max}} \tag{4.13}$$

Intuitively, this means that there must be at least N_{pts} grid points per wavelength so that the 'shape' of the wave can be adequately represented. Figure 4.1 shows how a sampled sine interpolated with different order interpolators approximates the exact sine with increasing number of sampling points. Figure 4.2 shows the interpolation error with increasing N_{pts} for linear, quadratic and quartic interpolation schemes. Interpolation error is defined as $e = \int_0^1 (y(t) - y_i(t))\ \mathrm{d}t$, where $y(t)$ is the target function (the sine in this case) and $y_i(t)$ is the interpolated function. At least from a discrete signal interpolation point of view it is seen that if the interpolation is of order greater than linear this approximation will converge faster. In particular note that interpolating quartically with 10 points is roughly equivalent to interpolating quadratically with 18 points. This point will prove important in later discussions.

In practice N_{pts} is often taken somewhere between 6 and 15 points per grid, depending on the order of the method of solution, which is determined numerically by analyzing how different grid

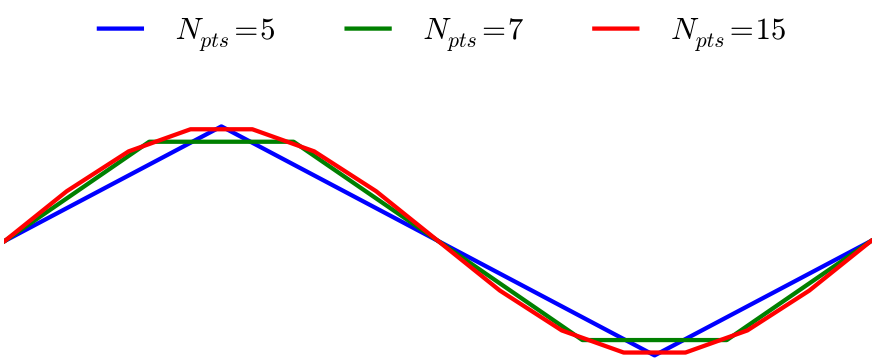


FIGURE 4.1. Linear interpolation of a sine wave with increasing number of points (N_{pts}). Note how $N_{pts} = 7$ completely misses the peaks.

spacings and time-steps resolve the solution to known problems. The slowest wave speed (β) is taken usually, approximately, as the shear wave speed at the topmost layers of the model ($\beta = V_s$). This is only an approximation because it disregards the speed of surface waves which can be slightly lower than the speed of shear waves. Surface waves, in fact, are dispersive meaning that their speed will depend on the wavelength (or frequency) of the wave. But computing the actual speed of surface waves for a complex geological model can be a challenge. It is usual, then to take N_{pts} in the higher range of acceptable values and use V_s as reference wave-speed.

The time-step of the simulation must be taken to satisfy the Courant–Friedrichs–Lewy (CFL) condition (Courant, Richard; Friedrichs, Kurt; Lewy, 1956). This condition states that the chosen time-step multiplied by the fastest rate at which information can travel (fastest wave speed) should be less than the grid spacing. For the seismic case, the compressional waves at the greatest depth modeled are the fastest possible speed at which information can travel.

$$\mathrm{d}t \leq \frac{\mathrm{d}r}{V_p} \tag{4.14}$$

Where $\mathrm{d}r$ is the smallest spacing between two points in the mesh in any direction, V_p is the speed of compressional (P) waves. The CFL condition places an upper bound on the time-step size for the simulation based on the chosen grid spacing. The time-step might be further constrained due to other reasons such as material non-linearity or numerical scheme stability conditions.

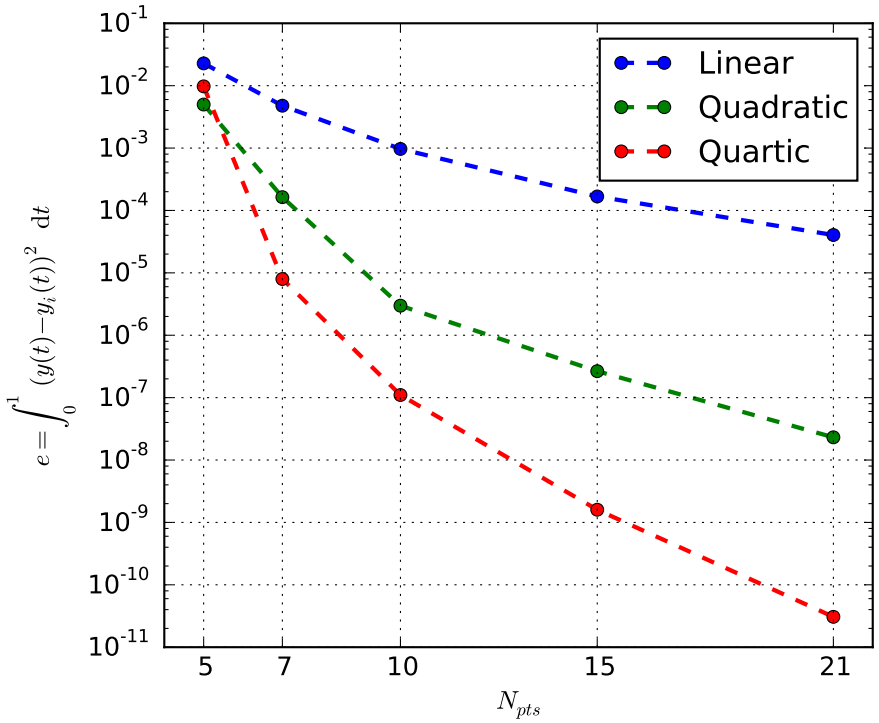


FIGURE 4.2. Sine-wave interpolation error for varying number of points and integration schemes. Doubling interpolation order can yield up to two orders of magnitude error decrease. Alternatively, doubling integration order allows the use of sample spacing twice as large.

The preceding analysis holds for the case of linear elastodynamics. As soon as there is any non-linearity in the problem, the convergence of the system should be carefully studied solving the problem at ever decreasing time-steps and grid spacings until convergence is obtained. Such convergence is not guaranteed. Furthermore, consistency, whether the approximate solution tends towards the exact solution as grid spacing decreases, is also not guaranteed.

4.3. Earthquake source characterization

Earthquakes are the shaking of Earth's surface due to seismic waves produced by the spontaneous, brittle rupture and slippage of the crust along planes of weakness known as faults. The driving forces behind earthquakes are the tectonic motions of the plates that compose Earth's crust. These relative motions produce accumulation of stresses in the crust and, in particular, along faults. Simplistically, when the shear strength of the fault is exceeded by the tectonic

stress, slippage occurs which might generate mechanical waves which propagate to the surface and around the planet.

Faults are commonly idealized as zero-thickness slip planes within a continuous medium. In reality, faults have a thickness where the inelastic deformations of crustal rock occurs. Furthermore, there are cases where it is difficult to establish a precise location for a fault since the motion and slipping manifests itself as plastic deformation and crack-growth over a zone of large extent. Nevertheless, the idealization of a zero-thickness slipping plane has proven adequate for most applications involving computations of shaking at a distance away from the fault.

The relative motion, or slipping, on either side of the fault is the result of a dynamic rupture process. This process is determined by the state of stress along the fault plane prior and during the rupture, the fault roughness characteristics, mechanical properties of surrounding crustal materials and previous rupture histories. Simulation of the rupture process by using first principles-based computations is called *dynamic earthquake modeling*. The seismic wave-field produced by these simulations can be then compared with recorded motions as a way of assessing the model validity. Alternatively, these models can be used to come up with new models for scenario earthquakes.

As an example, Peyrat, Olsen and Madariaga (Peyrat et al., 2001) computed a dynamic earthquake rupture model for the Landers 1992 models, by inversion of near-field accelerograms using the dynamic rupture model as forward problem. Schmedes and Archuleta (Schmedes et al., 2010) carried out 315 numerical simulations of dynamic ruptures in order to come up with a realistic rupture model generator (Schmedes et al., 2013), which was later adopted as an important ingredient in the UC Santa Barbara broadband platform for strong ground motion simulation (Crempien and Archuleta, 2015).

On the other hand it is possible to take the inverse approach: to use recorded motions obtained at seismic stations throughout the planet to *infer* what that rupture process might look like. This is called *inverse modeling*, because it uses the concept of wave propagation theory (forward model) to test several possible rupture scenarios and infer the most *plausible* one. The result of inverse model is a *kinematic earthquake model* which describes the rupture process without regards to the dynamic equilibrium equations which produced it.

Inverse modeling can be applied to different measurements of earthquake induced motions, as long as there is a theory to provide a *forward model*, i.e. a model that can predict the response based on the source. Seismic stations all around the world (or at tele-seismic distances) can be used to constrain location, centroid moment tensor, and, depending on earthquake size, rupture model for low-frequencies. Measurements of the co-seismic static displacement field through GPS or satellite based techniques can also be used to obtain models of earthquake ruptures. Finally, near-field measurements of strong ground motions carry information on high-frequency components of the rupture. All these datasets can be combined into so-called joint inversions which, allegedly, provide a rich view of the rupture.

The theory of finite-fault inversion by Ji, Wald and Helmberger (Ji et al., 2002a), and the later discussion of the arising complexities (Ji et al., 2002b) have been a very successful approach at kinematic source modeling from tele-seismic data. This approach was recently shown to be very succesful at modeling the 2014 Napa earthquake (Dreger et al., 2015), later this model was used successfully along with a 3–D model of the crust to predict ground motions at nearby sites (Rodgers and Pitarka, 2015).

This dissertation focuses on kinematic earthquake models. Seismic motions will be simulated using a-priori knowledge of earthquake occurrence and characteristics. These earthquake models might be realistic models stemming from seismic inversions, dynamic earthquake models, or simple scenario earthquake modeling, i.e. postulating a plausible earthquake source in order to assess its effects on soil-structure systems.

Throughout this work faults are idealized as zero-thickness surfaces, commonly flat planes, where slipping occurs during an earthquake. This means that the displacement field due to an earthquake is discontinuous through the fault surface, the discontinuity 'jump' in displacement is called the slip vector ($\Delta \boldsymbol{u}_s$) and is typically variable throughout the fault surface.

Figure 4.3 shows the parametrization chosen for a simple planar fault that might represent a small earthquake or be an element of a bigger one. Strike is measured clockwise from the X axis, dip is measured from the horizontal, and rake is measured clockwise from the horizontal. The rake vector shows the movement of the footwall (material points under the fault) relative to the hanging wall (material points above the fault). A strike-slip fault will have a dip of 90° and a 0°

rake, while a reverse fault might have a 45° dip and a 90° rake. The alignment of the X or Y axis with geographic directions is application specific.

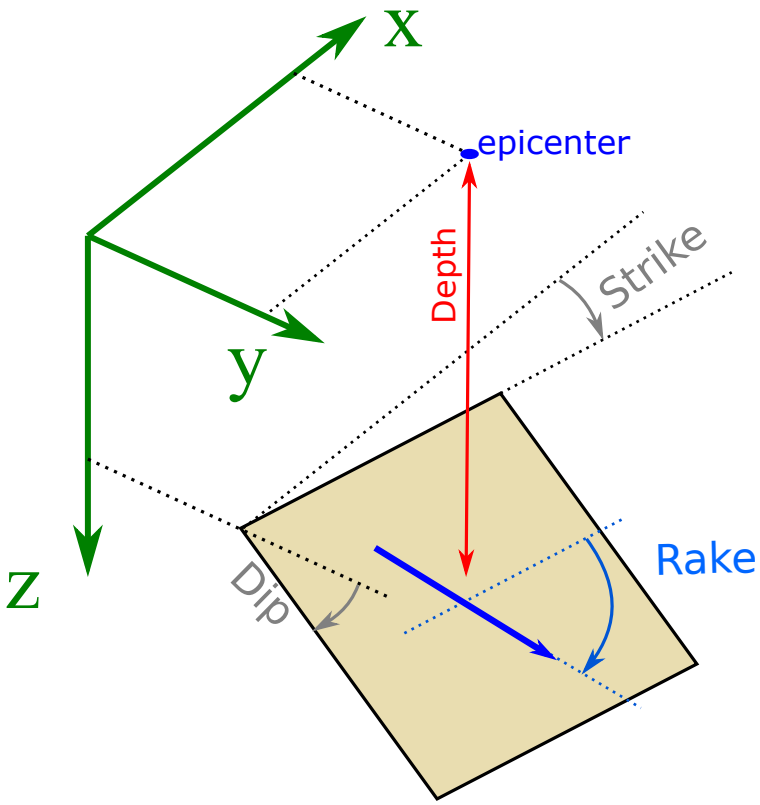


FIGURE 4.3. Fault geometry for a simple planar fault.

Consider a planar fault with a uniform slip vector across it, $\Delta \boldsymbol{u}_s$, and an area A with any closed shape. A result from elasticity theory is that the moment tensor (Scholz, 2002) associated with this situation is given by:

$$M_{0ij} = \mu \left(\Delta u_{s\,i} n_j + \Delta u_{s\,j} n_i \right) A \tag{4.15}$$

Where $\Delta u_{s\,i}$ is is the i-th component of the slip vector $\Delta \boldsymbol{u}_s$, n_i is the i-th component of the normal to the fault surface, A is the surface area of the fault and μ is the shear modulus of the crust nearby the fault. Please note that $\Delta u_{s\,i}$ is a function of time, since the displacement cannot occur instantaneously. This idea will be expanded on later.

The seismic moment (energy release or work done by the earthquake) is the tensor norm of this tensor, $M_0 = \mu \| \Delta \boldsymbol{u}_s \| A$. From this measure of energy Hiroo Kanamori, in his famous 1977

paper (Kanamori, 1977), defined the moment magnitude scale as

$$\log M_0 = 1.5M_w + 9.1 \tag{4.16}$$

Where M_0 is in $[\mathrm{Nm}]$.

In the case of a general, curved fault surface with slip vector as a function of space, one may equally define a moment density tensor as

$$m_{0ij}(\boldsymbol{x}) = \mu \left(\Delta u_{si}(\boldsymbol{x}) n_j(\boldsymbol{x}) + \Delta u_{sj}(\boldsymbol{x}) n_i(\boldsymbol{x}) \right) \qquad \text{, for } \boldsymbol{x} \in A \tag{4.17}$$

Where now the slip vector and the fault normal are both functions of space. The seismic moment for this extended fault will be the sum of the contributions of the sub-faults of which its composed:

$$M_{0ij} = \int_A \mu \left(\Delta u_{si}(\boldsymbol{x}) n_j(\boldsymbol{x}) + \Delta u_{sj}(\boldsymbol{x}) n_i(\boldsymbol{x}) \right) \mathrm{d}A \tag{4.18}$$

A large, extended fault (possibly curved or composed of several segments), can be approximated by *discretizing* it into many planar sub-faults as shown in Figure 4.4. The premise is that if it is possible to accurately compute earthquake motions for a simple plane fault, a bigger fault can be discretized into simple faults, with uniform slip on them, and the superposition principle invoked to compute the motions due to it. In this case the total moment is computed as a summation over the moment contribution of all sub-faults, using the appropriate crustal properties at the location of each sub-fault.

If the points where motions are to be determined are far away from the source, when compared to the relative size of the fault, then the plane fault may be further approximated by considering it to be a point source. A point source has no spatial extent but does have an orientation as defined by the slip plane it models and the direction of slipping. Conceptually, a point source can be defined a the limit as the fault extents tend to zero while keeping the total energy release on the fault constant.

A forcing function for such a case can be defined as follows (Burridge and Knopoff, 1964)

$$F_i(\boldsymbol{x}) = g(t - t_0) M_{0ij} \nabla \delta(\boldsymbol{x}_j - \boldsymbol{x}_{0j}) \tag{4.19}$$

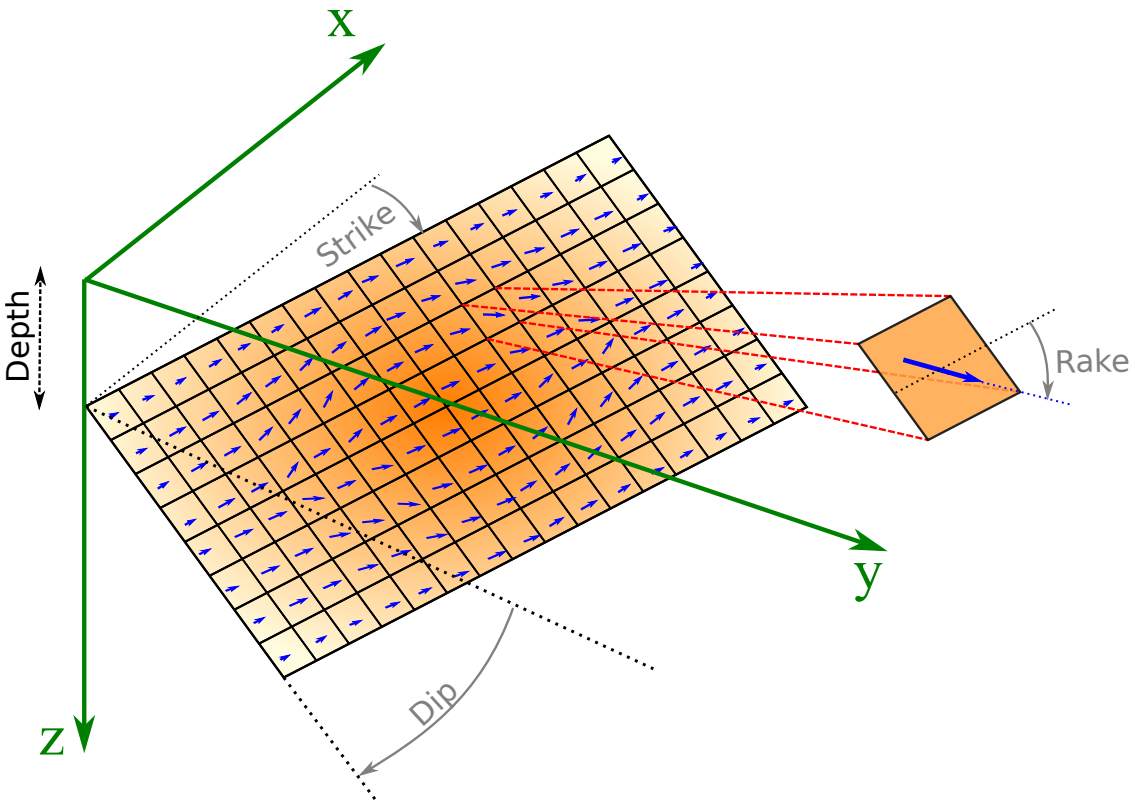


FIGURE 4.4. An extended fault discretized in many planar faults

Where $\delta(\boldsymbol{x})$ is the 3–D scalar Dirac delta function defined at point $\boldsymbol{x}$, $\nabla\delta(\boldsymbol{x})$ its gradient, $\boldsymbol{x}_0$ is the hypocenter of the earthquake, M_{0ij} is the final moment release, and $g(t)$ is called the moment release function (also source time function) with t_0 the start time of the function. This unitless function, $g(t)$ controls the release in time of moment or slip, which cannot be considered to occur instantaneously, has value of 0 at the start of the event ($t = 0\,[\mathrm{s}]$) and a unit value at the end (which might be modeled to be $t = \infty\,[\mathrm{s}]$). The shape of this function directly controls the frequency content of the resulting motions (Brune, 1970), in general short-duration functions emphasize the high frequency portion of the spectrum while the opposite occurs for long-duration source time functions. The next section lists a few functional forms that might be used for $g(t)$.

To completely describe an extended fault one needs to define the extents of each sub-fault (length in strike and dip directions), the orientation of said fault (strike and dip angles), the slip vector (rake angle and displacement magnitude), the time evolution of the slip vector (moment release function or its derivative the moment rate function) and the time of initiation of motion

for each sub-fault. The moment release function might have additional parameters requiring specification.

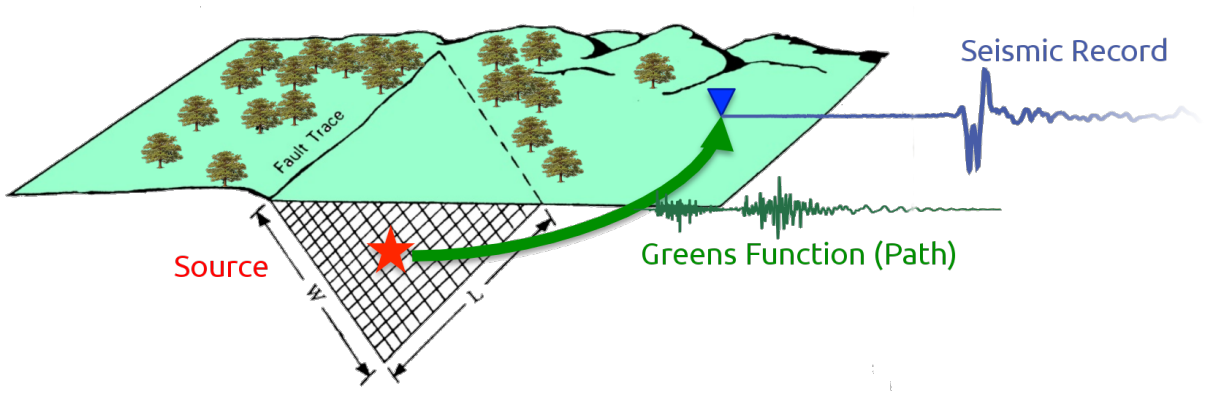


FIGURE 4.5. A fault is discretized into various point sources. The contributions of each sub-fault are summed at the site of interest.

4.4. Postulating new earthquake scenarios

Postulating a plausible earthquake scenario is by no means a trivial task. Any earthquake scenario that aims at representing the actual geological threat to a structure will have to be devised based on extensive knowledge of the local seismic setting. This means knowing the past history of the fault, what size earthquakes its capable, the crustal properties near the fault, how much slip might be accumulated and released, geometry and depth of the seismogenic layer. What follows is a review of the kind of parameters that need to be defined, and some ideas on how to come up with them.

Performance-based earthquake engineering (PBEE) requires definition of different levels of seismic loading in order to compute metrics of performance. Size of earthquakes relates to the magnitude of the event which has a certain probability of exceedence given the probabilistic seismic hazard analysis (PSHA) of the site. In earthquake scenario modeling, once a magnitude, location (distance), depth, and faulting style parameters are selected (from the PSHA or GMPEs) there are several of additional parameters which are left free to the modeler, but which must be selected in an informed way so as to produce a credible earthquake scenario.

First such parameters is deciding what is the fault rupture extent given the magnitude. Apart from the magnitude itself, the rupture extent is directly related to the event duration, frequency

content, slip amount, and co-seismic deformation field extents and shape. The scaling of rupture extent with magnitude is region, and even fault, specific (Mai and Beroza, 2000). Moreover, the scaling law might change depending on the magnitude of the event (Scholz, 2002; Stock and Smith, 2000). The reason for this is that a small or medium sized earthquake might be completely contained within the seismogenic layer (Earth's schizosphere), so it might be expected that rupture *area* relates in some way to the magnitude. Large earthquakes exhaust the seismogenic layer in depth leaving the strike-direction extension of the rupture as the only possibility of growth. Thus for large earthquakes it is expectable to have a scaling law that relates rupture *length* with magnitude. From a dynamic rupture propagation perspective, even the evolution of the rupture will depend on the current size of the rupture.

After the geometry of the rupture is chosen, it is necessary to define a discretization of the fault into sub-faults or elementary faults. This discretization will depend on the bandwidth of the resulting motions which needs to be computed. A broad discretization will be needed if only low frequencies are of interest (for very flexible structures like long bridges, tall buildings, or fluid system response to name a few) and will have to be fine if high-frequency components need to be resolved (for stiff systems like short buildings and bridges, nuclear power plants, tunnels, etc.). It is well known that the high-frequency content of seismological records is controlled in part by the roughness of faulting (Schmedes et al., 2013) and crustal heterogeneity. Therefore it is important to resolve these features at the source-modeling level if it is desired to accurately represent them.

Next it is important to devise a slip (moment tensor) distribution over the fault. Experience with inverse kinematic modeling and also dynamic fault rupture modeling shows that there are many way to distribute slip on a fault which are consistent with a given global seismic moment and faulting style. From an idealized point of view, unimodal slip distributions (like maybe a Gaussian distribution) is the most common one to model low-wavenumber features of the fault slip distribution while some stochastic process is chosen for high-wavenumber features. See (Schmedes et al., 2013) and (Crempien and Archuleta, 2015) for details on this. Nevertheless, multimodal events containing one or more asperities are not uncommon (Fortuño et al., 2014).

The nucleation point, or hypocenter, is the point on the fault where rupture begins. The centroid moment tensor (CMT) is the apparent point where, on average, all the energy seems to radiate from. The location of the hypocenter and CMT are one of the first scientific descriptions of the event which are available soon after the event since automated algorithms for their location have been deployed successfully worldwide. The relative position of the hypocenter with respect to the CMT hints at the mode of rupture: whether it is unilateral rupture or bilateral rupture. CMT and hypocenter coinciding in space suggests that the earthquake ruptured bilaterally, i.e. starting at the hypocenter and rupturing mostly concentrically outwards from there on. If there is an offset between both, then unilateral rupturing is suspected. In unilateral rupturing the rupture begins at the hypocenter and progresses into a specific direction. Unilateral ruptures tend to have more directivity effects which make shaking more intense in the direction of rupture. It is a modeling assumption whether the fault ruptures unilaterally or bilaterally, which may be based on experience from previous events on the fault.

Next, it is important to determine the speed at which rupture occurs or, similarly, the triggering times (t_0) of each sub-fault on the event. Typically it is suggested that the rupture speed is about 80% of the local shear wave speed. On the other hand, some earthquakes have been better explained by a super-shear type event, where rupture progresses at a speed faster than the local shear wave speed. In any case, the rupture speed can never exceed the speed of P-waves in the medium as this would break the causality principle for wave propagation. Super-shear events promote pulse-like motions along the directions of slip and tend to give shorter-duration higher-frequency-content motions than sub-shear ruptures. Again selecting a rupture speed must be based on previous evidence for the region.

Finally, to define each sub-fault, it is required to specify a source time-function. One very successful early suggestion for a time function comes from Brune (Brune, 1970), which derived a functional form which produced motions which preserved the high-frequency cutoff observed in recorded motions for small events. Other shapes have also been chosen to emphasize different portions of the response spectrum. Ultimately, for scenario modeling, it is a modeling assumption.

The preceding discussion only defines the earthquake event. The resultant motions will be further influenced by the geology they encounter as waves travel from source to site. Arguably, it is harder to define the geology in an acceptable manner. The next section will explore simple geological models to assess commonly encountered assumptions on motions used for engineering practice and to emphasize the importance of realistic geological models in the computation of ground response.

Earthquake-scenario modeling might be most useful if it is acknowledged that there is considerable uncertainty in all the parameters involved. Scenario modeling should then take into account this uncertainty and draw not one but several models randomly from the range of plausible values, defined formally through probability densities. Once the models are run and ground motions are obtained, it must be proven that the resultant site-specific motions cover the range of expected motions anticipated by the PSH analysis.

4.5. A study on wave propagation in layered a medium

When devising a model of geology from the almost total lack of information that is usual in the field, it is very common to assume a horizontally stratified layered arrangement of crustal materials. Layers might be composed of a uniform material or have smoothly varying properties with 'jumps' between layers. This section will numerically explore response of layered medium to earthquakes modeled as point sources, as well as the case of adding an intrusion of stiff material which breaks the layering. This concept of vertically propagating shear-waves will be shown to be at most an approximation of questionably accuracy, even in the presence of perfect layering, when near field motions are considered.

There are many reasons why a horizontal layering is a very convenient simplifying assumption. A critical reason is that horizontal layering will bend shear waves to the point that they become vertically propagating shear waves, greatly simplifying the input of earthquakes into models of site and/or structure. This happens because, just like light gets refracted when going through materials of different speed of light, seismic waves too get refracted at boundaries with sharp velocity contrast, following Snell's law. But since seismic waves are have different component body waves or phases, in addition to refraction, reflection and conversion of phases (P to S

and vice-versa) also occurs. Also, sharp interfaces act as wave-guides where waves get trapped as surface waves, most notably at the free surface.

There is only one documented case, the Lotung site (Borja et al., 1999), where truly vertically propagating shear waves were observed. In that case there was minimal vertical component recorded, but there were two horizontal components.

A layered model of geology might be accurate in two situations. First, on a flat site where the geology is composed of deposited soils and only for short scales (up to a few hundred meters). When only long periods are of interest. Indeed, at very large scales (tens of kilometers or more) Earth's material layout is essentially unidimensional (with depth), hence such models are very useful for global seismic monitoring. In the intermediate scale of a few kilometers to hundreds of kilometers, it is common to observe drastic changes in materials both horizontally and vertically. Frequencies of engineering interest (maybe 0.1 [Hz] to up to 25 [Hz]) involve wavelengths within this intermediate scale, therefore being sensitive to the changes in materials.

To show the response of a layered and intruded geology, a widely verified and validated software for high-performance simulation of seismic waves using finite differences, SW4 (Sjögreen and Petersson, 2011), will be used to simulate seismic motions. SW4 allows modeling of unbounded (visco-)elastic domains subject to arbitrary seismic sources in its interior. Reflections from model boundaries are eliminated by using the 'super-grid' technique (Petersson and Sjogreen, 2013). For domain sizes and grid spacings of engineering interest, these simulations are computationally costly and require the use a distributed memory parallel computer. In this case, the simulations were done using ESSI Computer at the University of California at Davis.

The basic model developed for this study is shown in Figure 4.6 and again in Figure 4.7a. It consists of a 1–D velocity structure, with S-wave speeds ranging from 500 [m/s] at top to 2500 [m/s] at 1000 [m] of depth, after which the speed is considered constant at 2500 [m/s]. For the P-Waves V_p is taken as $V_p = 2V_s$. The computational domain is a rectangular box with length of 10 [km], width of 5 [km] and depth of 5 [km], and a discretization of 10 [m] grid spacing is considered (leading to a maximum resolved frequency of 5 [Hz]. To study the effects on ground motions of earthquake depth, dip and geological layering, a reverse-faulting (dipping in the X

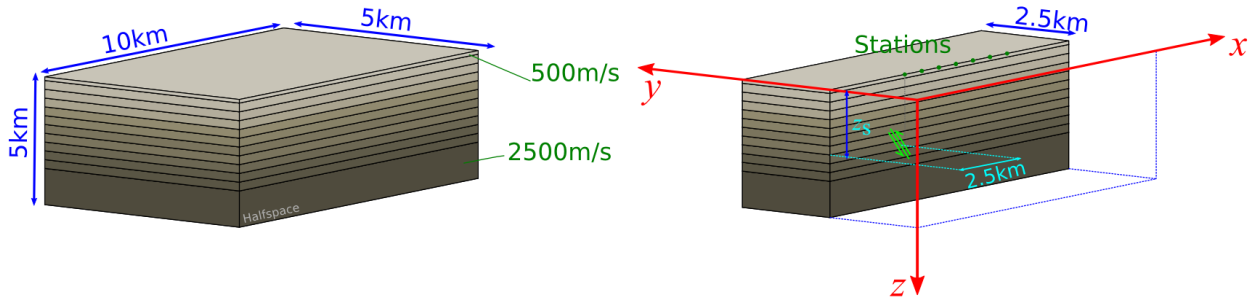


FIGURE 4.6. (Left) Not to scale, basic geometry of the area modeled. (Right) Coordinate system and location of earthquake source and receiver stations.

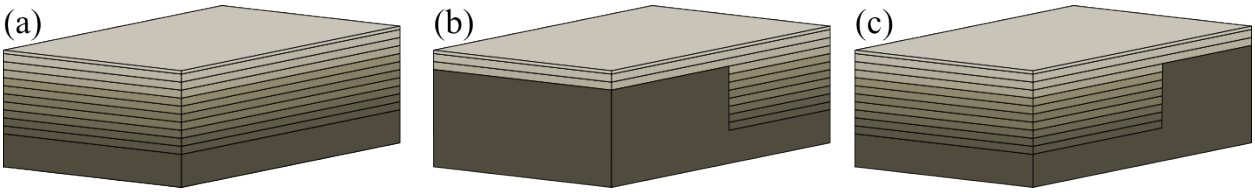


FIGURE 4.7. Material distributions (a) layered, (b) layered with intrusion on left of the domain, and (c) layered with intrusion to the right of the domain.

direction) double-couple source is placed within its interior at depths 550 [m], 850 [m], 1500 [m] and 2500 [m]; with dips of 30°, 45° and 60°.

Two additional models include a geological intrusion (which can also be viewed as a 'valley edge') is modeled as a block of hard material present at shallow depths. The feature occupies a side of the domain starting at a depth of 200 [m]. Figure 4.7b illustrates the intrusion located at left of the domain respectively (when looked from the $-Y$ plane), while Figure 4.7c shows the same for a right location of the intrusion.

The fault strike, slip and rake and moment magnitude completely define the moment tensor through 4.15. A simple Gaussian pulse, illustrated in Figure 4.8 is used as source time function. This functional form was chosen to simplify the analysis and interpretation of the wavefronts, reflections and refractions on layers and intrusion.

The Gaussian pulse is defined by

$$g(t) = \frac{\omega}{\sqrt{2\pi}} \exp\left\{-\frac{(\omega t)^2}{2}\right\} \tag{4.20}$$

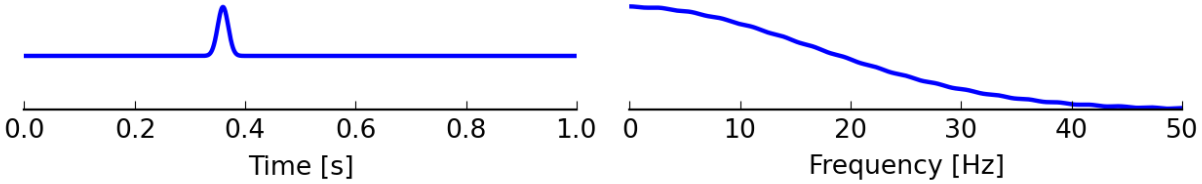


FIGURE 4.8. Gaussian pulse used in this work.

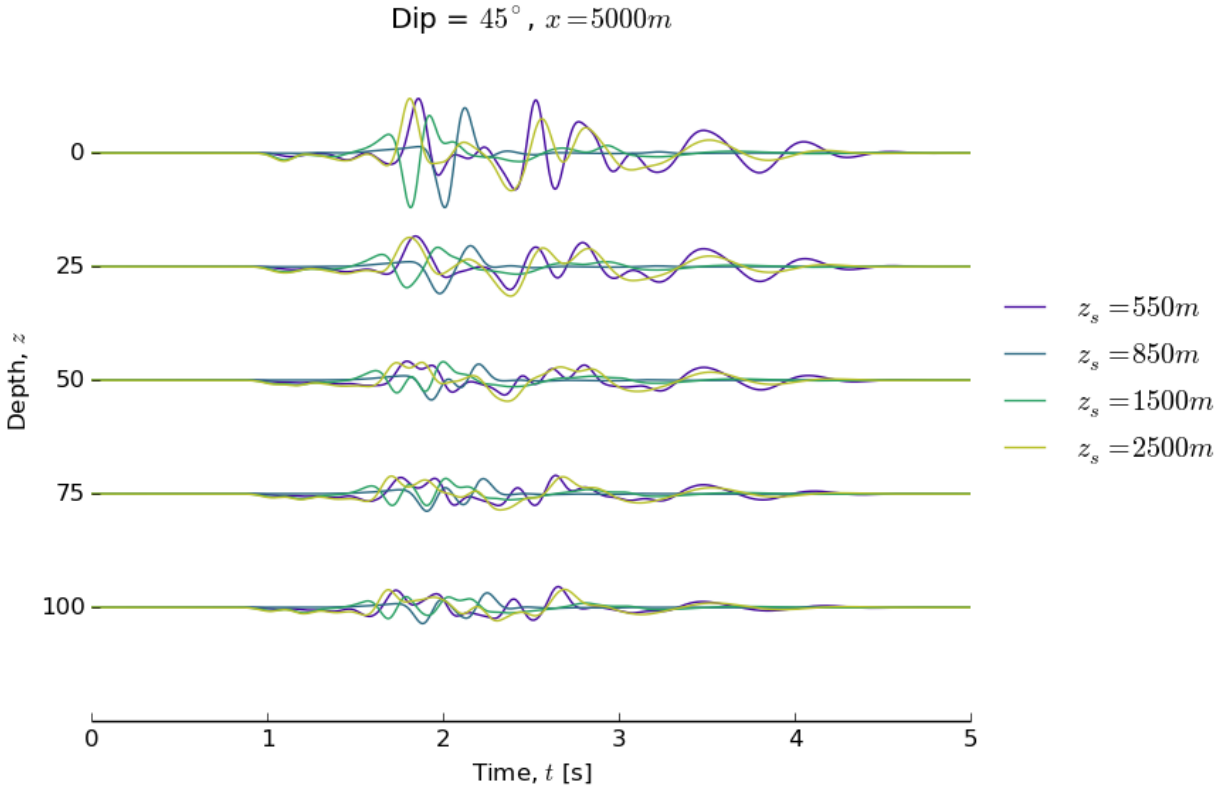


FIGURE 4.9. X-component of response in depth at the site.

The single parameter ω controls the width of the wave in time and frequency. Larger values lead to a narrower pulse in time, and a wider frequency bandwidth.

Figure 4.9 shows the horizontal displacements seen at a station in the location of the site (geometrical center of model, x = 5000 [m]) due to sources of 45° dip at varying depths. Figure 4.10 shows the vertical displacements for the same cases. To facilitate the interpretation of the trace shapes as contributions of surface and body waves to overall motion, the displacements in both figures have been normalized to the maximum value of either the vertical or the horizontal

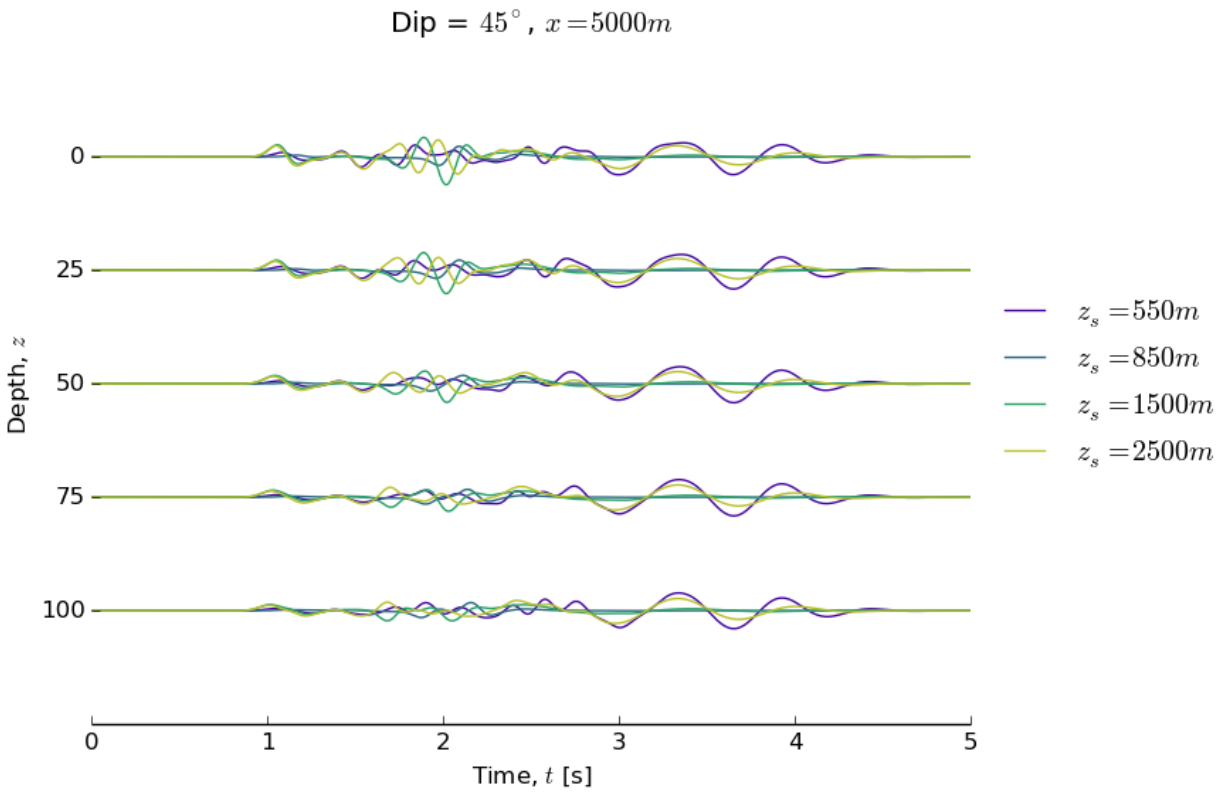


FIGURE 4.10. Z-component of response in depth at the site.

component (whichever is larger). This eliminates the dependency of amplitude with depth of source, but preserves the relative amplitude of horizontal to vertical components for analysis.

Figure 4.11 shows particle motion traces in the X-Z plane, showing time as varying color of the lines. At-a-glance it is obvious that layering cannot justify a vertically-propagating shear wave as local site ground motion. Is is only the deepest source that produces something approximately vertically propagating. Event though there still is a vertical component, this is due to an inclined S-wave field and not a P-wave as 1–D site response models suggest.

For the deep sources (z_s = 1500 [m] and z_s = 2500 [m]), the arrival of body waves is clear. At early times, blue color, particle motion is approximately radial from the source, indicating a P-wave. Next, in slighter green color, the arrival of S-waves is marked by a trace perpendicular to this previous one. Finally, for late times, yellow colors, the motions settle into the passage of Rayleigh waves. In this case, maximum motions are dominated by body waves and it is apparent that S-waves become curved until they assume a vertically propagating motion (SH-waves).

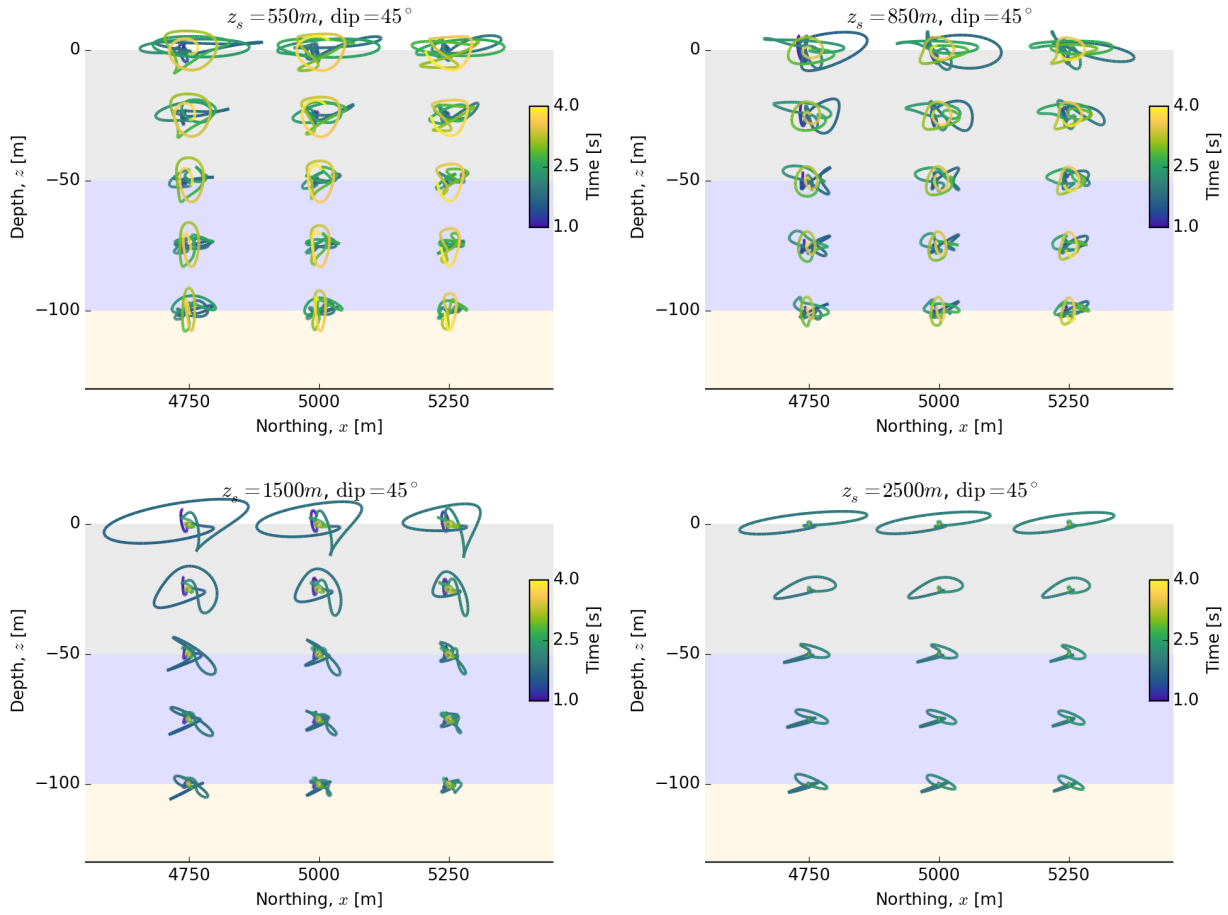


FIGURE 4.11. Traces of particle motions around the site of interest for different source depths for the layered geology. Line-color shows time and layering is also shown for reference. Note that only the deep source can be considered to produce approximately vertically propagating S-Wave.

At shallow source depths ($z_s = 550\,[\text{m}]$ and $z_s = 850\,[\text{m}]$), this pattern is broken. Initial motion horizontal motions for the shallowest source are due to P-Waves propagating horizontally along the layers from the shallow source. Due to source geometry (45° dip) P waves propagate vertically and horizontally while S waves propagate at 45° angles, therefore 'tunelling' P-waves along the strata. In this case, motions due to body waves are of about same amplitude as the surface waves.

Shown in Figures 4.12 and 4.13 are the results for the case when a hard rock intrusion is modeled at the left and right of the domain for a 45° dipping fault respectively. When the

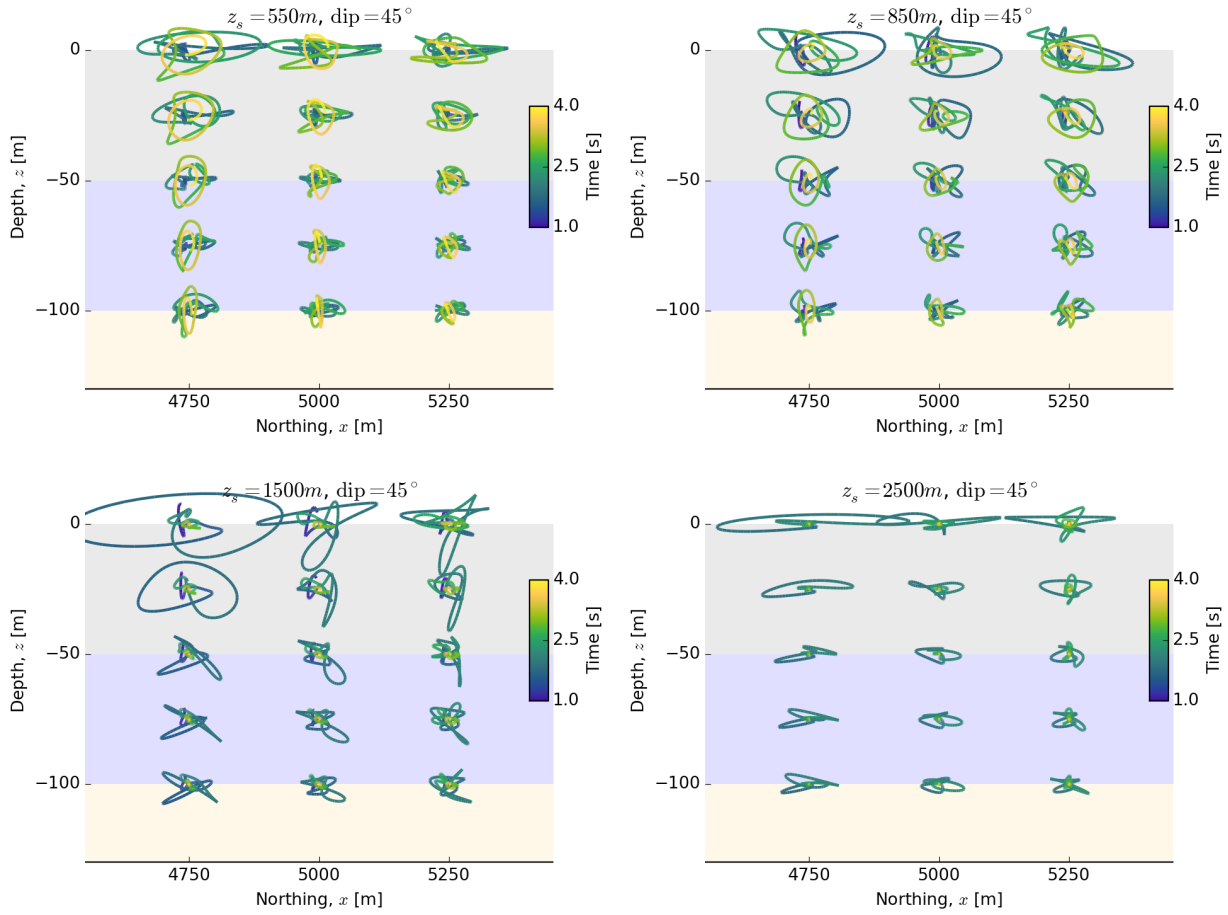


FIGURE 4.12. Particle motion traces for the case of intrusion located at the right of the domain.

intrusion is located on the left, all sources are on hard material. When its on the right we have two sources on softer material and two on hard.

For the intrusion on the right of the domain (Figure 4.13) we see little effect of the intrusion before the waves reach its vicinity. The effect of the intrusion is to bend the body waves generating reflections and refractions at the interface. The effect is most apparent for the z_s = 2500 [m] source, which is located just under the soft layers but is shallow enough to go through a significant portion of the softer domain.

The intrusion on the left alters the propagation pattern more severely. In that case, the source is located within the hard material. Response to the deepest source appears to be the least affected with respect to the purely layered case. This is because, due to the location of the

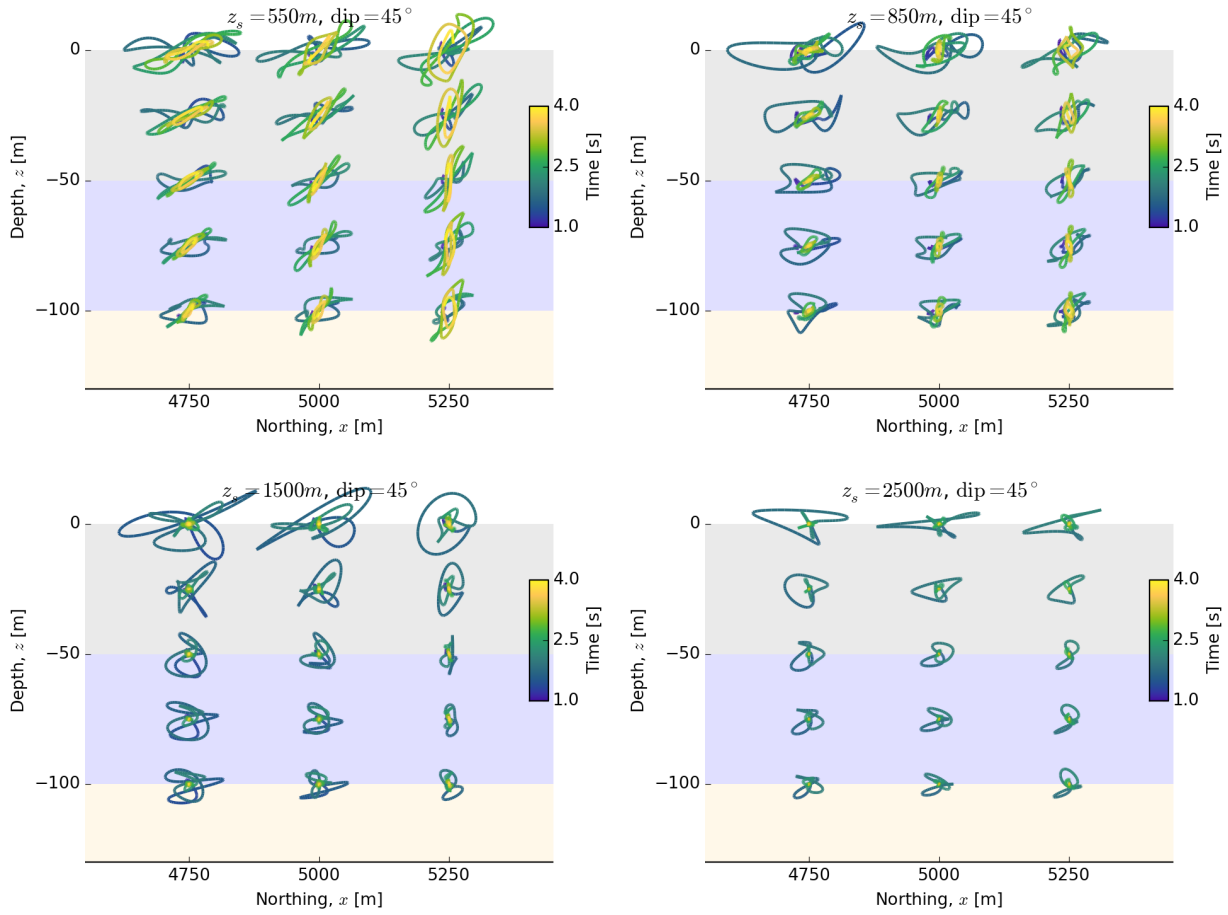


FIGURE 4.13. Particle motion traces for the case of intrusion located at the left of the domain.

fault, the wave-field mostly by-passes the intrusion vertical face. All other source depth feature complicated, fully 3–D motions which have been altered heavily by the intrusion.

By comparing layered case to the cases with intrusions it can be seen that in the layered case once the waves leave the source (or any particular point) they travel outward never to return. That is, any given place in the domain is only visited once by a particular wave phase and its conversions. In the cases presenting the intrusion, reflections at the intrusion boundaries arrive at a later time back to the source or the site. This is an important feature of heterogeneous geology, that there are reflections and trapped wave in the geology. A very important case is the so-called 'basin-effect' in which waves are trapped within a basin of soft material surrounded by hard material but, furthermore, they become focused by the curved edges of the material.

This study involving a highly idealized scenario, shows that if anything it would be coincidental that 1–D site conditions would be observed in the field. Empirical evidence backs up this claim by the scarcity of 1–D (even 2-D horizontal) site records. Additionally, at engineering scales, sharp stiffness contrast due to non-layered geology can further distance the shaking conditions from the 1–D site condition.

A rational ESSI analysis for system subjected to near field sources, therefore, requires that the full 3–D wave-field be characterized for correct assessment of input motions. Furthermore, realistic soil behavior is highly dependent on volumetric deformations. These deformations are not captured by the classical assumption, and can be captured by a complete ESSI simulation.

4.6. Development of Seismic Wave Fields for ESSI Modeling

This section explores the numerical conditions under which a high-performance fourth-order finite difference code for seismic modeling, henceforth the 'seismic' code, can be coupled successfully with a general purpose parallel non-linear finite element simulator, the FEM simulator, through the use of the domain reduction method (DRM). The approach taken consists in modeling a simple homogeneous half-space subjected to a single double-couple point-source to generate motions. DRM will be used to input these motions into the FEM simulator for an equivalent model of the domain, and the response compared at a control point on the surface. The seismic code used is SW4 (Sjögreen and Petersson, 2011) developed at Lawrence Livermore National labs, while the FEM simulator will be the Real ESSI Simulator developed at the University of California Davis. Both are high-performance parallel programs highly regarded in their respective domains of application.

As originally proposed by Bielak (Bielak et al., 2003), the DRM input motions can be generated using a different method to compute the seismic wave field than the FEM code used to model site and structure. The rationale being that both methods will be approximating the same equations of elastodynamics and should both converge to the same solution as grid spacing tends to get smaller. What was implicit in that seminal work, but not explored or tested, is the effect of using different methods with possibly different orders of convergence, and how this affects convergence of the overall method.

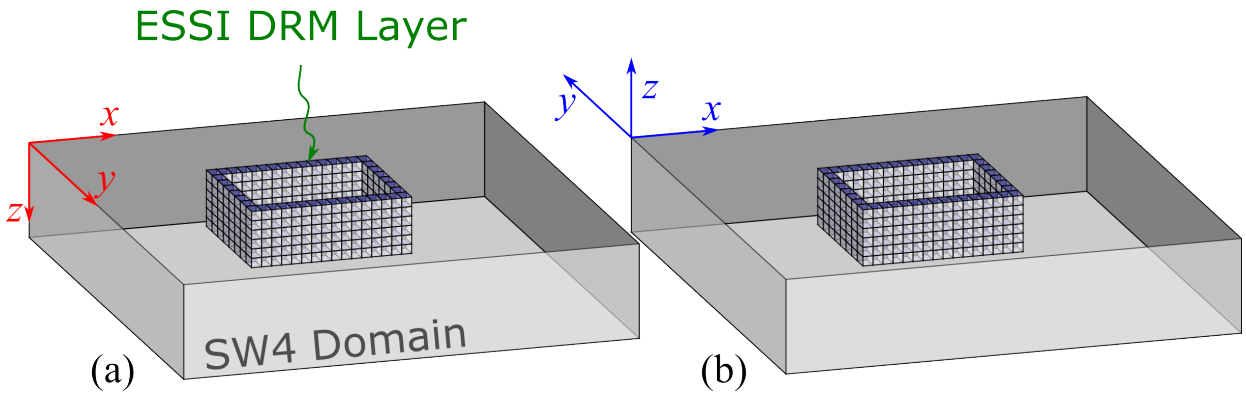


FIGURE 4.14. Model layout and coordinate system origin and orientation for (a) SW4 (b) ESSI Simulator

The advantage of using different codes is that it is possible to choose a 'seismic modeling' code which is better suited and optimized for simulating earthquakes and then use the DRM to input the resulting motions into a code which is more suitable for modeling of non-linear soil and structural behavior in the chosen site. For example, SW4 has very convenient features for the input of double couple sources and also for extended sources, a task which would be much harder to achieve in a civil-engineering oriented code such as Real ESSI Simulator. It is important to use the proper tools for modeling task, both for efficiency and credibility of the results used for design.

When solving the elastodynamic equations, it is expected that different solution schemes will yield different solutions for the same problem. Even if the seismic code's finite difference grid points coincide spatially with the FE nodes, the different mathematical transformations involved in advancing the solution and the different orders of approximation will no-doubt lead to some degree of disagreement on the nodal values. When using the seismic code as DRM input into the FEM code it is expected that this disagreement will manifest itself in two ways: first, that the solution at a common control point within the DRM domain and on the seismic domain will differ and, second, consequently there will be a portion of the wave-field that will not be absorbed at the DRM boundary leading to outgoing motions which need to be damped out. Again, it is expected that both the difference in motions and the outgoing wave-field will diminish with decreasing spatial and temporal discretization.

TABLE 4.1. Mesh information for DRM models used with RealESSI simulator.

h, [m]	**Number of Nodes**	**Number of elements**
20.0	28,577	3,200
10.0	78,141	9,000
5.0	295,829	35,000

Figure 4.15 illustrates the general layout of the DRM boundary elements with respect to the SW4 model for the domain used in this study. The seismic model consists on a single material elastic box of size $8\,[\text{km}] \times 4\,[\text{km}] \times 4\,[\text{km}]$, various values of the grid size are chosen ($h = 10\,[\text{m}]$ or $20\,[\text{m}]$), a point double-couple source is placed at $(2\,[\text{km}],\, 2\,[\text{km}], 2\,[\text{km}])$ such that it represents a reverse fault with a 45° dip. The center of both models coincide. SW4 uses a right-handed coordinate system with z-axis positive downwards, while the FE meshes developed[1] for ESSI simulator use a right-handed system with z-axis upwards. Both models x-axis' and origins are made to coincide. Thus, the transformation from SW4 coordinates to Real ESSI Simulator coordinates is:

$$x_{\text{SW4}} = x_{\text{ESSI}} \qquad y_{\text{SW4}} = -y_{\text{ESSI}} \qquad z_{\text{SW4}} = -z_{\text{ESSI}} \tag{4.21}$$

Figure 4.15 shows the construction of the DRM model used in this study. It consists of an internal domain (within the DRM boundary) of dimensions $200\,[\text{m}] \times 200\,[\text{m}] \times 40\,[\text{m}]$ with discretization $\mathrm{d}x = \mathrm{d}y = \mathrm{d}z = h = 5\,[\text{m}]$, $10\,[\text{m}]$ or $20\,[\text{m}]$. The elastic properties throughout both SW4 and Real ESSI Simulator domains are such that the speed of P-waves is $V_p = 2000\,[\text{m/s}]$, the speed of S-waves is $V_s = 1000\,[\text{m/s}]$ and the density is $\rho = 2000\,\left[\text{kg/m}^3\right]$.

For the absorbent layer 4, 8 and 16 elements are used with Rayleigh damping ratios of $\xi = 0,\, 0.05,\, 0.1$. The RealESSI meshes are made up of second-order 27-node bricks with 27 Gauss-integration points. Table 4.1 summarizes the number of elements and nodes for the RealESSI Simulator meshes used.

The RealESSI meshes are generated using *gmsh* (Geuzaine and Remacle, 2009) and results are visualized using the custom visualizer plugin `visitESSI` (Abell, 2015) for the VisIt (Childs et al., 2012) high-performance visualization tool.

[1] This is for convenience when using the meshing program *gmsh* to develop the mesh.

Figure 4.16 shows the visualization of the displaced RealESSI mesh for $h = 20\,[\mathrm{m}]$ (left) and $h = 10\,[\mathrm{m}]$ (right) at time point $t = 3.40\,[\mathrm{s}]$. No damping was used at the absorbing boundary at this point. This instant in time corresponds to the end of the arrival of the P-wave and, it can be observed, that DRM is working to eliminate the outgoing motions as the absorbing boundary has little or no displacement. Furthermore, the $h = 10\,[\mathrm{m}]$ mesh seems to be doing a better job at absorbing the out-going motions. Figure 4.17 shows the same situation for time point $t = 3.93\,[\mathrm{s}]$. At this time the S-wave is coming into the domain. It can be seen that the $h = 20\,[\mathrm{m}]$ domain is producing large out-going motions while the finer domain is still handling the out-going motions.

Figures 4.18a and 4.18b shows the trace plots for the control point at the center of the domain for both SW4 results as well as the results with DRM when using a DRM domain with discretization $h = 20\,[\mathrm{m}]$ and $h = 10\,[\mathrm{m}]$ respectively.

P-waves are faster than S-waves, in this case, by a factor of 2. Since wave resolvability for a given time-step size is proportional to wave speed, P-waves will be better resolved than S-waves. This manifests itself in the fact that initially, for P-wave arrivals, DRM works at capturing out-going motions and later, for S-waves, some out-going motions escape the boundary. These out-going motions are not absorbed by damping since damping is not applied at this point, therefore, reflecting off the model boundaries back into the domain. This why there is oscillatory motion, corresponding to energy trapped in the system, observed after $t = 4\,[\mathrm{s}]$. Note that the oscillatory motion is mainly seen in the x-component and not the others.

To mitigate these trapped waves, the absorbent boundary is assigned Rayleigh damping of varying intensities. Rayleigh damping can be applied in many ways. From (Tafazzoli, 2012) several lessons can be drawn regarding how to design the absorbent boundary for maximum efficiency. The following is a summary of these lessons:

- Increasing in-absorbent-boundary damping has the general effect of reducing the amplitude of waves reflected at the domain boundary back into the internal domain.
- Sharp damping contrasts produce additional reflected waves at the DRM/absorbing boundary interface.
- Gradually increasing the damping ratio with increasing distance to the DRM boundary alleviates the reflection issue while retaining a similar damping efficiency.

- Increasing the thickness in elements of the absorbent boundary also increases the damping efficiency.
- When selecting two frequencies to provide for specification of Rayleigh damping, the best frequency is not related to the 'natural' frequency of the soil stratum as is commonly assumed in practice and some research. The best frequency is related, instead, to the frequency of the out-going waves.

Additionally, inspired by the formulation of the supergrid method (Petersson and Sjogreen, 2013), it may also prove beneficial to consider changing the stiffness of the elements in the absorbing boundary with distance to the DRM layer. This is left for future research.

In this study a uniform value for the damping ratio assigned to the Rayleigh damping coefficients is used. Figure 4.19 shows the effect that this additional damping has on DRM models with different sizes. The beneficial effect of this damping in capturing the energy leaking out from the DRM boundary is apparent.

Perfect matching of the motions obtained with SW4 and RealESSI simulation with DRM modeling was not achieved. A key reason for this is that the SW4 simulations were done at $h = 20\,[\mathrm{m}]$ while varying the mesh size for the RealESSI simulations. This means that SW4 motions had to be interpolated between grid spaces when the grids did not match. SW4 provides only 'nearest' neighbor interpolation for requested output stations, so an improvement on this is needed if better matching is expected. Alternatively, it is possible to achieve better agreement by decreasing the discretization on both domains.

From these experiments it is possible to extract the following design considerations when seeking to couple DRM-based finite element simulations with other forms of seismic modeling:

- Size of the DRM domain is irrelevant for the 'free-field' problem.
- Relative order of accuracy of the Finite-element mesh and the code that produces seismic input is important when designing mesh sizes for both the finite-element and the seismic simulation.
- Matching motions perfectly might result in expensive computations, beyond what is needed due to physical and numerical constraints for the propagation problem alone.

- Out-going motions due to mismatch need to be absorbed outside the DRM domain by some method.

Ultimately, the purpose of using the DRM is as the enabling technology allowing the rational modeling of perturbations of the free-field model in order to reduce the cost of jointly modeling complex site and structure response along with the seismic wave propagation problems. In such a case, out-going motions will be unavoidable and have to be dealt with efficiently. In any such study it will be very important to demonstrate that the DRM motions and forces developed agree with the free-field model as a basic way of showing the adequacy of the numerical implementations involved in the modeling effort.

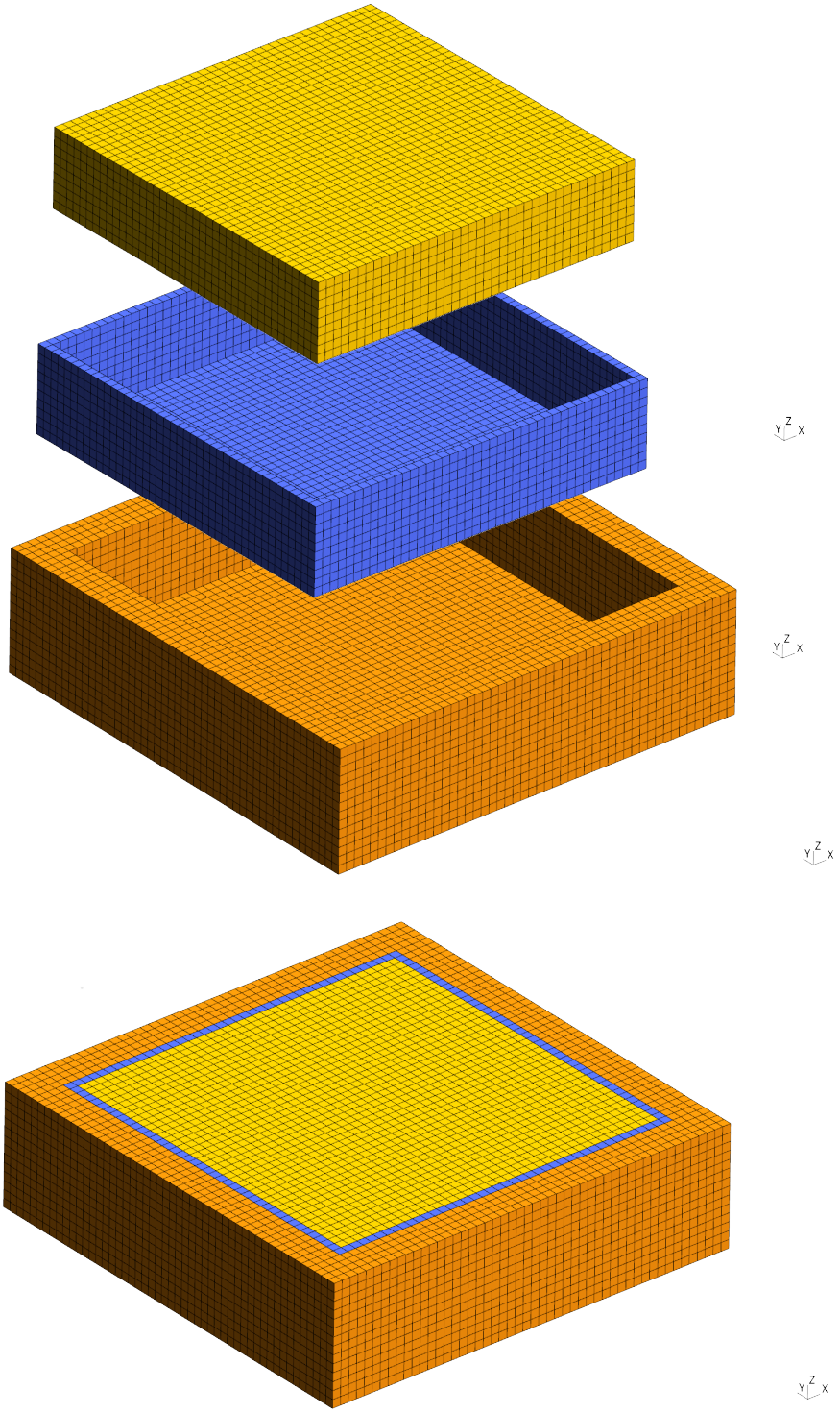


FIGURE 4.15. Free field DRM model for verification of SW4 and ESSI Simulator coupling. From top, internal domain, DRM boundary, absorbent element layer, complete model.

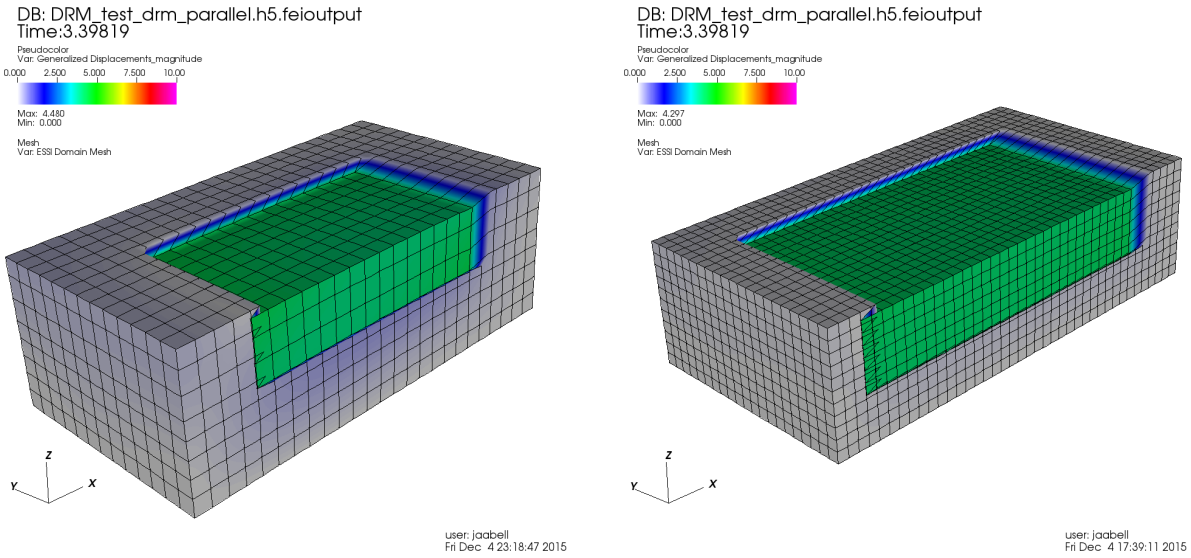


FIGURE 4.16. Visualization of the DRM solution at time $t = 3.40\,[\mathrm{s}]$ for DRM mesh sizes of $h = 20\,[\mathrm{m}]$ (left) $10\,[\mathrm{m}]$ (right), SW4 mesh is $h = 20\,[\mathrm{m}]$ for both cases. Color shows magnitude of displacement vector. Arrival of P wave is correctly resolved on both meshses.

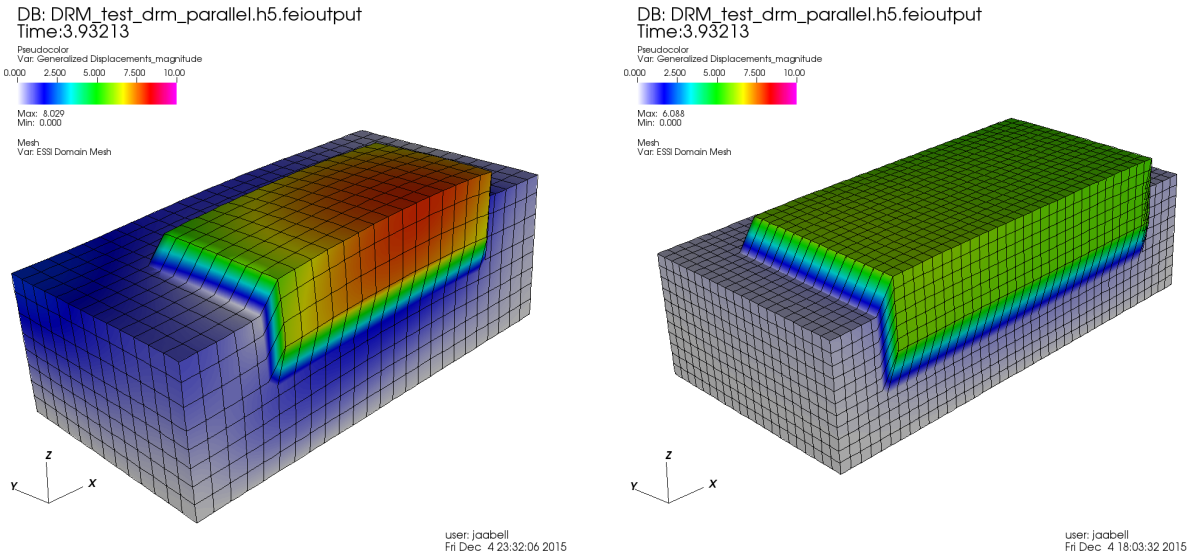
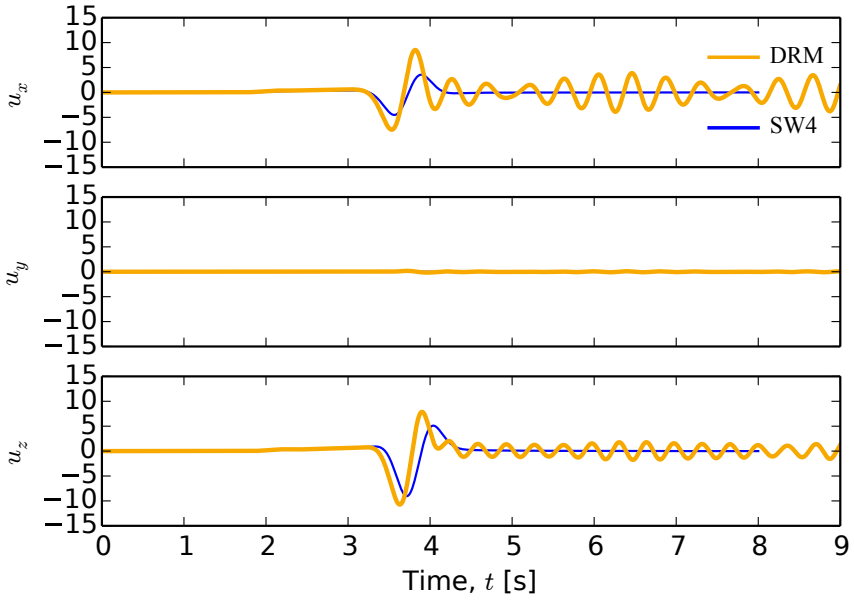
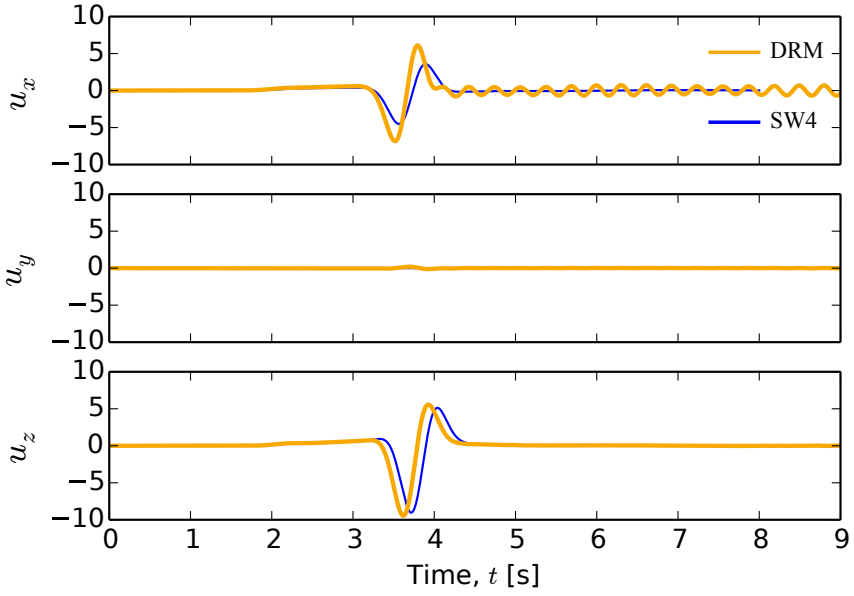


FIGURE 4.17. Visualization of the DRM solution at time $t = 3.93\,[\mathrm{s}]$ for same setup as Figure 4.16, with color showing magnitude of the displacement vector. Arrival of S wave is better resolved on the $h = 10\,[\mathrm{m}]$ mesh but not on the $h = 20\,[\mathrm{m}]$ mesh as can be seen by looking at the out-going motions. SW4 mesh is $h = 20\,[\mathrm{m}]$ for both cases



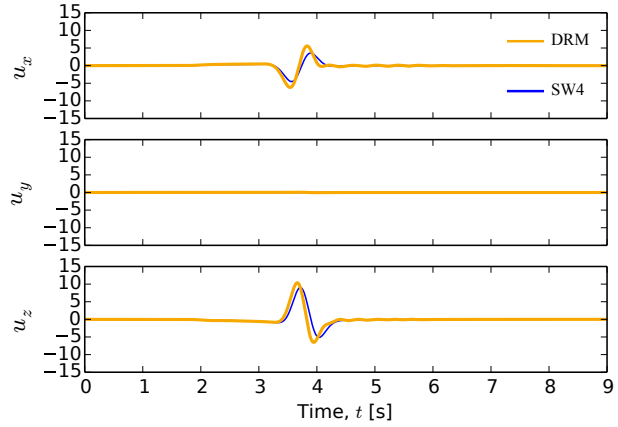
(A) 20m grid spacing



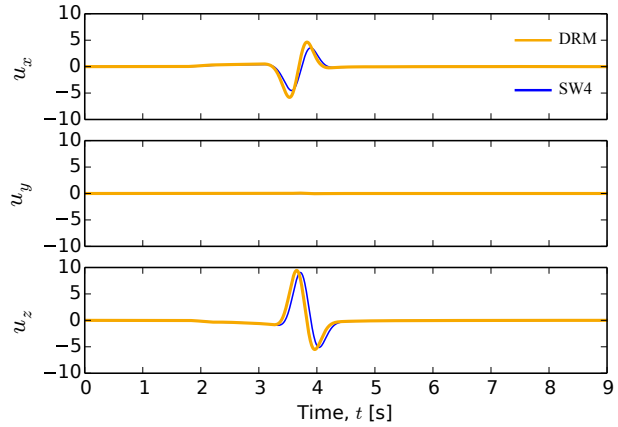
(B) 10m grid spacing

FIGURE 4.18. Generated motions at the center node ($x = 4000m$, $y = 2000$, $z = 0$), blue shows SW4 motions generated with an $h = 2\,[\mathrm{m}]$ grid, and orange shows motions obtained with RealESSI using DRM and variable mesh size.

(A) $h = 20\,[\mathrm{m}]$



(B) $h = 10\,[\mathrm{m}]$



(C) $h = 5\,[\mathrm{m}]$

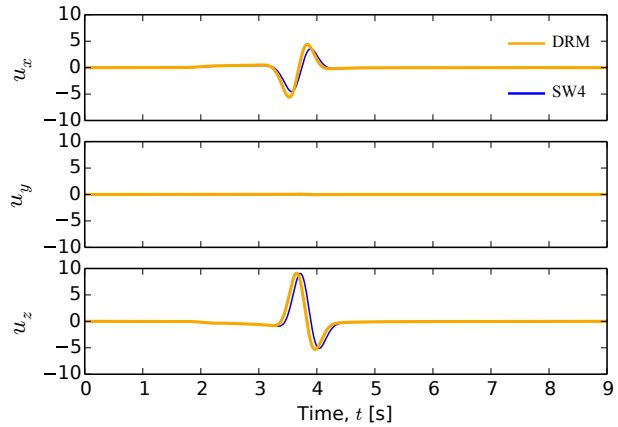


FIGURE 4.19. Results from computing motions with ESSI Simulator within the DRM domain for different ESSI mesh sizes h and Rayleigh damping with $\xi = 0.1$. All SW4 motions were computed for $h = 20\,[\mathrm{m}]$

CHAPTER 5

Response of a Soil-Structure System to 3–D and 1–D Representations of the Seismic Wave-Field

5.1. Problem Statement

The method for generation of seismic motions presented in the previous chapter will be used to produce seismic wave field to be applied to a model of a nuclear power facility founded on non-linear soil.

Figure 5.1 shows a drawing of the nuclear power plant (NPP) in plan and elevation, featuring both the containment as well as the auxiliary building. The structure has a 1:1 plan aspect ratio. Reinforced concrete walls with thicknesses ranging from $0.4\,[\text{mm}]$ to $1.6\,[\text{mm}]$ are spaced every $12.5\,[\text{mm}]$ withing the auxiliary building. The containment building consists of a $40\,[\text{mm}]$ diameter and $40\,[\text{mm}]$ high cylinder capped by a dome, with $1.6\,[\text{mm}]$ of wall thickness. The NPP is has a $3.5\,[\text{mm}]$ thick continuous foundation slab.

The facility will be founded on a layered half-space identical to that used in the previous chapter. Surface shear wave velocity is $V_s = 500\,[\text{mm}]$ for the first $50\,[\text{mm}]/$, and the compressional wave speed is twice that. A non-linear elasto-plastic constitutive model will represent the non-linearity of soil in the neighborhood of the NPP. The soil model will present typical degradation properties for granular materials, and purely deviatoric response.

Two sets of motions will be generated to test the main hypothesis of this work. The first consists of a 45° dip reverse fault source located at a horizontal distance of $2\,[\text{km}]$ and a depth of $2\,[\text{km}]$ with a corner frequency of $f_0 = 6.2\,[\text{Hz}]$, meant to produce an approximately 1–D condition seismic field in the vicinity of the NPP. The second source is a shallower one, located at a depth of $850\,[\text{mm}]$ and with a corner frequency of $f = 8\,[\text{Hz}]$, meant to produce a wave field which deviates from the ideal 1–D condition.

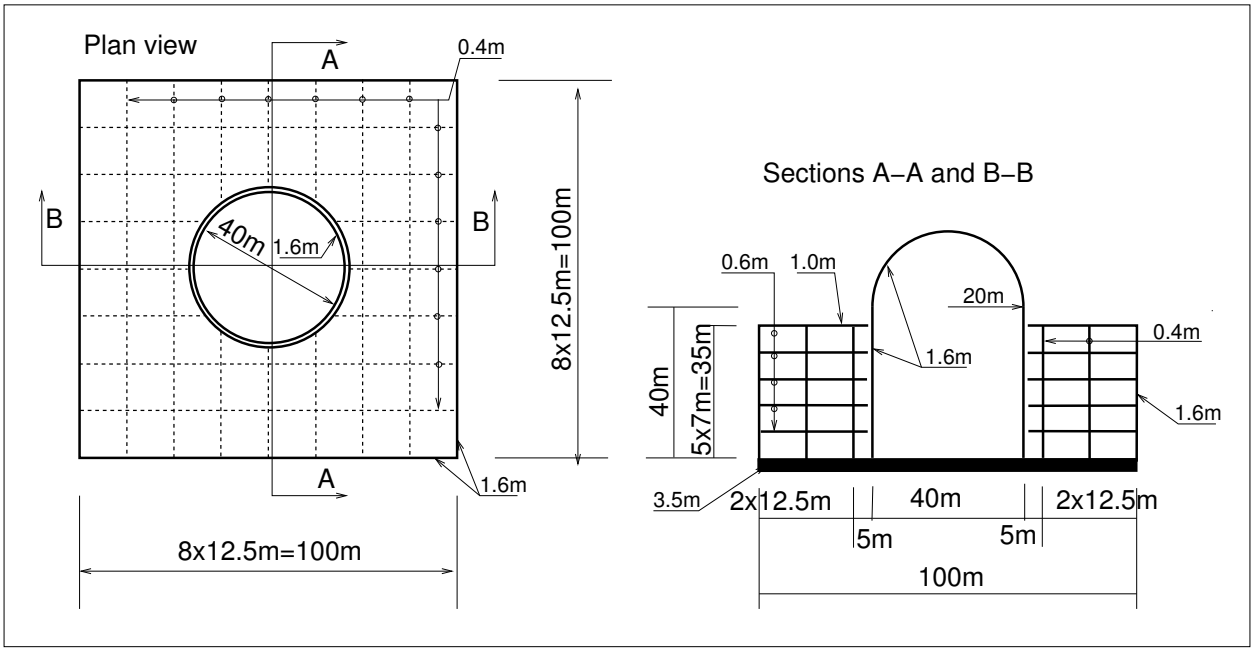


FIGURE 5.1. Nuclear power plant geometry

Linear and non-linear response of the NPP will be computed for these two motions. Additionally, full 3–D results will be compared with a 1–D site approach in which the three components of the free-field surface response are de-convolved in depth and input into the model using the DRM.

5.2. Finite-element mesh of the NPP and neighboring soil

The `gmsh` (Geuzaine and Remacle (2009)) meshing program was used to construct a threedimensional geometrical model of the NPP and produce a mesh composed of hexahedral finite elements for soil and foundation slab and quadrilateral shell elements for the walls and slabs of auxiliary building as well as the containment building.

Figure 5.3 shows the different components of the mesh created. The discretization of the model is controlled by the discretization of the free-field, further controlled by the accuracy of wave propagation. For a maximum frequency of $10\,[\mathrm{Hz}]$[1] and the selected surface shear wave velocity, a discretization of size $\mathrm{d}x = 5\,[\mathrm{mm}]$ is needed. Considering the aspects presented in the previous chapter and other geometrical constraints of the mesh, a mesh size of $\mathrm{d}x = 3.2\,[\mathrm{mm}]$ is

[1] actually inputs are of less bandwidth than this

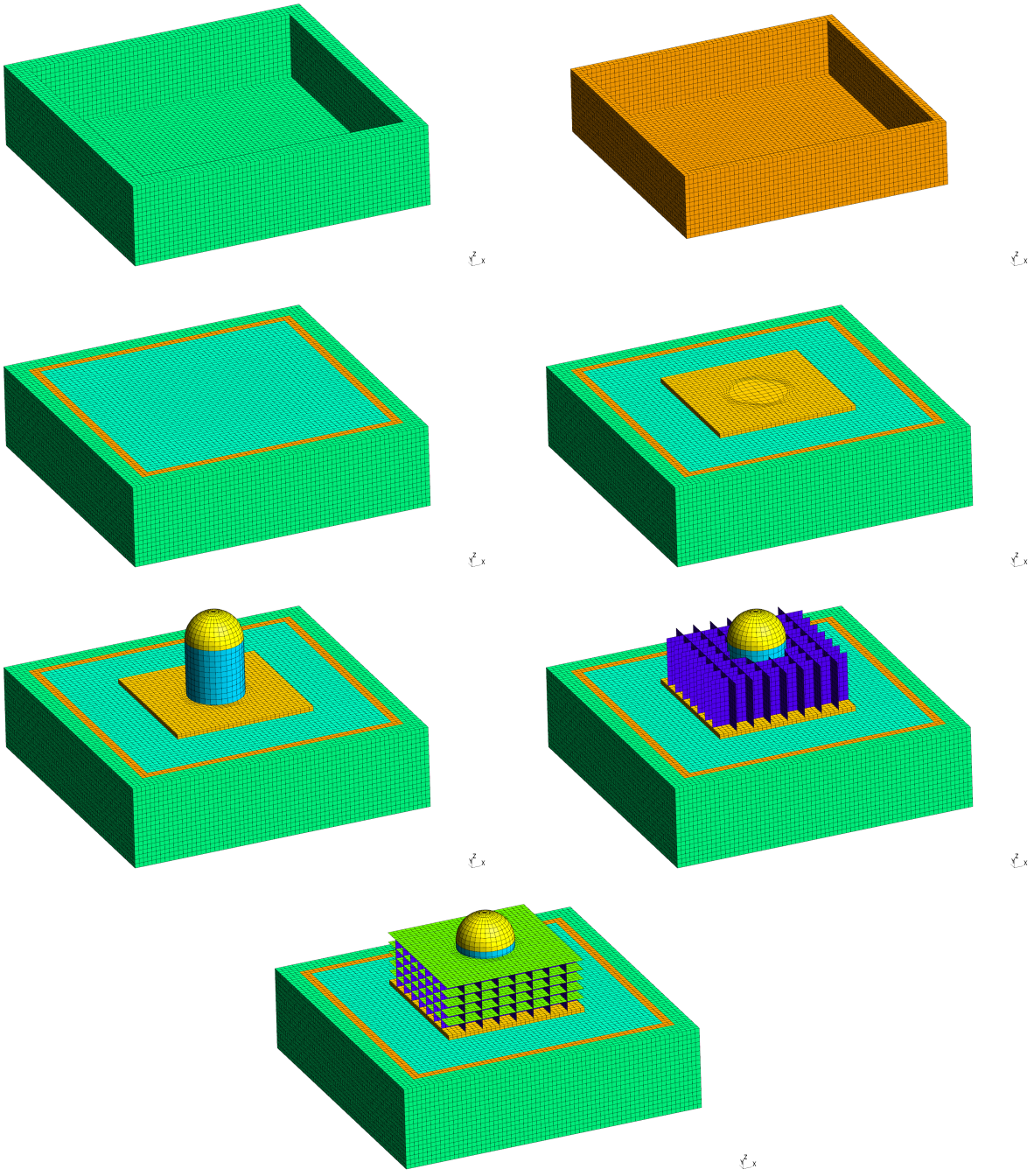


FIGURE 5.2. FEM mesh of the site.

selected for the soil. This mesh size determines the mesh size of the NPP, but wave speeds on concrete are higher than those of the soil, therefore this selection is appropriate.

The resulting mesh consists in 106,165 nodes , of which 9,728 define 6 degrees-of-freedom (three translations and three rotations) and 96,437 define 3 degrees-of-freedom. A total of 100,736 elements are defined, consisting of 31,824 shells and (68,912) eight-node bricks. 37,632 of the continuum elements will contain non-linear material representing the soil, while the rest use linear materials representing the DRM layer, absorption layer and foundation slab. This results in 551,296 Gauss integration points of which 301,056 are non-linear material points.

It is important to mention that all shear walls are extended into the foundation slab to capture the rotational stiffness of the slab. Not doing so would result in rotations at the base of the walls for out of plane wall motion, with the implication that wall bending moments at the base of the walls are zero for out of plane motion. In reality, walls are built continuous with the foundation slab and are able to develop bending moments.

The ANDES shell formulation (Bergan and Felippa (1985); Alvin et al. (1992); Felippa and Militello (1992); Felippa and Alexander (1992)) was followed and implemented in Real-ESSI as a high-performance linear quadrilateral element. This element formulation is able to capture the vibrational dynamics of shells in high accuracy at the cost of forfeiting the concept of displacement interpolation functions. Given the thickness in shear walls for auxiliary building and containment building, a linear model for concrete behavior suffices.

The DRM layer consists of one element in thickness for the DRM input and a 5-element thick absorption layer with high-damping elements.

A Rayleigh damping model (see equation A.33) is used to assign damping to linear parts of the domain, as well as some viscous damping for the soil. Damping ratio is set at 20% of critical for absoption layer elements, 5% for soil and 2% for structure and foundations.

5.3. Non-linear soil model

A simple Drucker-Prager with Armstrong-Frederick hardening constitutive model is chosen as the material model for the soil. The chosen parameters are a shear wave velocity of $V_s = 500\,[\mathrm{m/s}]$, mass density $\rho = 2000\,\mathrm{kg/m^3}$, $phi = 5^\circ$, $h_a = 1000$, $c_r = 973$. These parameters lead to very early yielding of the material while the ultimate strength is a friction angle of $\phi = 30^\circ$. The material behaves as non-dilatant due to a Von-Mises type plastic flow direction.

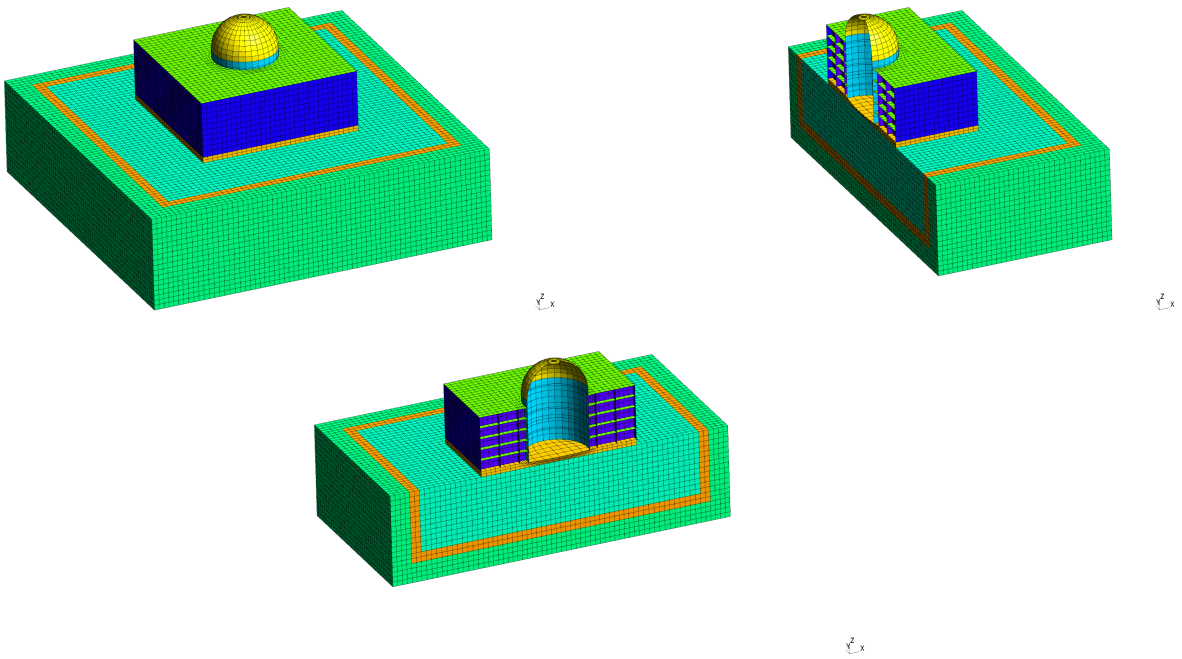


FIGURE 5.3. FEM mesh of the site and NPP featuring, in orange, the layer of DRM elements.

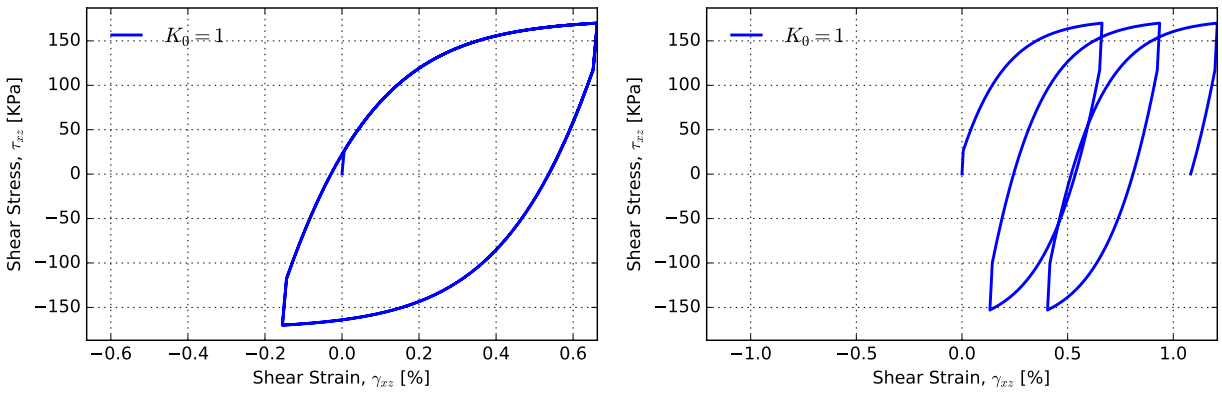


FIGURE 5.4. Response of material to symmetric and unsymmetric loading showcasing the ratcheting due to the Bauschinger effect

Figure 5.4 show the cyclic response of the material to stress-controlled symmetric cycles and unsymmetrical cycles. The Bauschinger and ratcheting effects due to the non-linear hardening laws are also shown. Figure 5.5 shows the stiffness degradation curve for the chosen parameters.

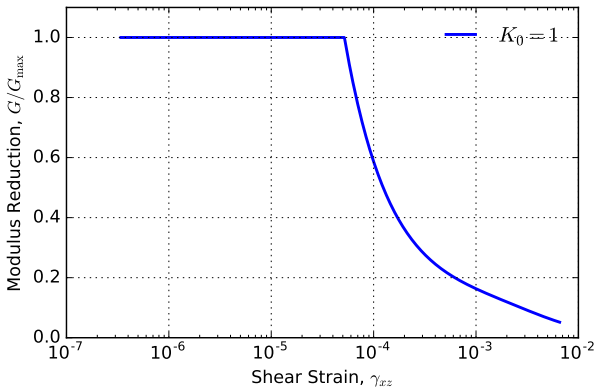


FIGURE 5.5. Stiffness degradation response of the material

5.4. Earthquake source time-function

Capturing an adequate frequency content for the resulting motions is very important for a realistic analysis using point sources. Brune ((Brune, 1970, 1971)) derived the shape of the source time function based on physical considerations of the available stress to drive the near-fault acceleration of a rupturing fault. This source time functional shape, shown in Equation 5.1 when used to drive the elasto-dynamic equations resulted in a frequency spectrum which mached the observed high-frequency spectrum.

$$g(t,\, t_0\, \omega_0) = \begin{cases} 0 & t < t_0 \\ 1 - \exp\left\{-\omega_0 (t - t_0)\right\} \left(1 + \omega_0 (t - t_0)\right) & t \geq t_0 \end{cases} \tag{5.1}$$

In Equation 5.1 t_0 is the rupture start time and $\omega_0 = 2\pi f_0$ and f_0 is the so-called 'source corner frequency' which is related to the magnitude of the event and the 'stress-drop' $\Delta\sigma$ which is the change in shear stress accross the fault which drives the acceleration on the fault. The source corner frequency is given by

$$f_0 = 4.9 \times 10^6\, V_s \left(\frac{\Delta\sigma}{M_0}\right)^{1/3} \tag{5.2}$$

Where f_0 is in Hertz, V_s is the shear-wave velocity near the fault and is in $[\mathrm{km/s}]$, $\Delta\sigma$ is in bars, and M_0 is the source moment in $[\mathrm{dyne \cdot cm}]$. In SI units, f_0 in Hz, V_s in $[\mathrm{m/s}]$, $\Delta\sigma$ in Pascals, and

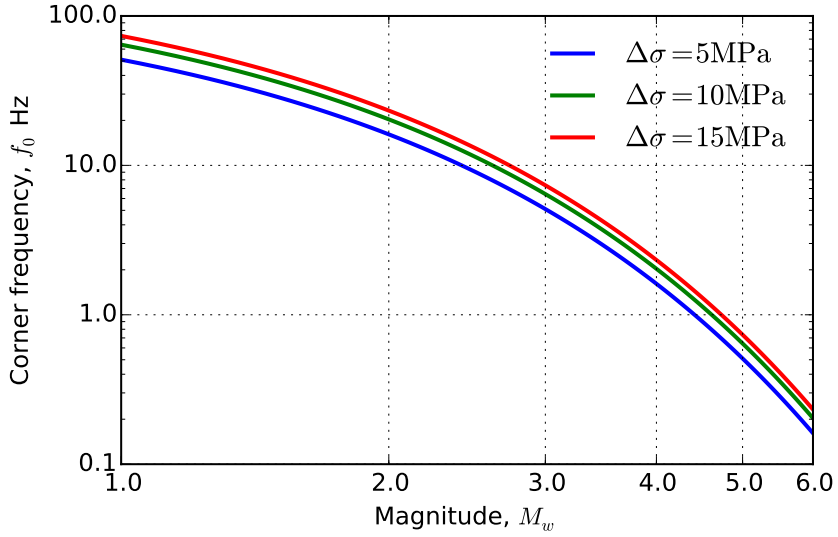


FIGURE 5.6. Corner frequency vs. moment magnitude for different values of the stress drop.

M_0 in $[\mathrm{N}\cdot\mathrm{m}]$ the equation is:

$$f_0 = 4.9\times10^{-9}V_s\left(\frac{\Delta\sigma}{M_0}\right)^{1/3} \tag{5.3}$$

Figure 5.6 shows plots of the corner frequency with magnitude for a Brune-type source. In this study a $M_w = 3.0$ source with a stress drop $\Delta\Sigma = 10\,[\mathrm{MPa}]$ is used to produce a corner frequency of $f_0 = 6.2\,[\mathrm{Hz}]$.

The practical disadvantage of the Brune source function is that it has a discontinuous second derivative at $t = t_0$, which produces high-frequency response in SW4, a fourth-order code. To improve this situation, SW4 provides a smoothed version of the Brune source time function, which is three times differentiable at $t = t_0$. The `BruneSmoothed` function replaces the beginning of the function for times $\omega_0(t - t_0) < \tau_0$ by a version which will provide a 5 times differentiable function at $t = t_0$, and match the Brune function and its derivatives at $\omega_0(t - t_0) = \tau_0$. The chosen

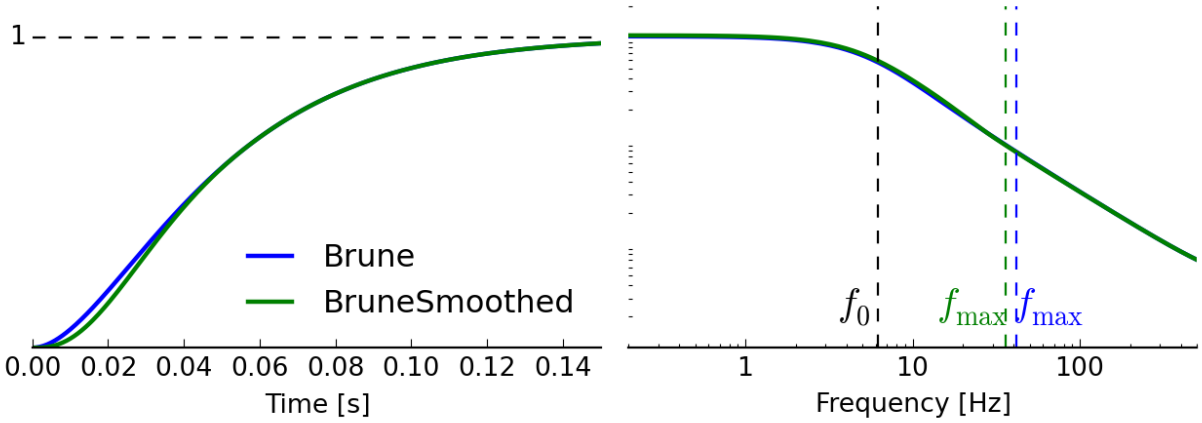


FIGURE 5.7. Time and frequency plots for a Brune and BruneSmoothed source time functions both for $f_0 = \frac{\omega_0}{2\pi} = 6.2\,[\text{Hz}]$

function is

$$g(t,\, t_0\, \omega_0) = \begin{cases} 0 & t < t_0 \\ 1 - \exp\{-\omega_0(t-t_0)\}\left[1 + \omega_0(t-t_0) + \frac{1}{2}(\omega_0(t-t_0))^2 + \ldots\right. & \\ \left. -\frac{3}{2\tau_0}(\omega_0(t-t_0))^3 + \frac{3}{2{\tau_0}^2}(\omega_0(t-t_0))^4 - \frac{1}{2{\tau_0}^3}(\omega_0(t-t_0))^5\right] & 0 < \omega_0(t-t_0) < \tau_0 \\ 1 - \exp\{-\omega_0(t-t_0)\}\,(1 + \omega_0(t-t_0)) & \omega_0(t-t_0) \geq \tau_0 \end{cases} \tag{5.4}$$

where $\tau_0 = 2.31$. Figure 5.7 shows the time and frequency plots of the Brune and `BruneSmoothed` functions for the chosen corner frequency $f_0 = 6.2\,[\text{Hz}]$. The maximum frequency $f_{\max}$, a measure of the bandwidth of the source-time-function, is defined as the frequency at which the amplitude of the Fourier spectrum of the source-time-function drops below 5% of its peak value. For the Brune source with $f_0 = 6.2\,[\text{Hz}]$ the value of $f_{\max}$ is $41.35\,[\text{Hz}]$ while for the `BruneSmoothed` it is $35.94\,[\text{Hz}]$. So the `BruneSmoothed` function has a slightly smaller $f_{\max}$ than the Brune function, but can still be expected to provide ample bandwidth to excite the upper layers of the domain.

5.5. Generated surface motion and 1–D Site deconvolution

Using the smoothed Brune source time function as slip function for the seismic events, two SW4 simulations are executed for the two different source depths and corner frequencies ($z_s = 2000\,[\mathrm{m}]$ with $f_0 = 6.2\,[\mathrm{Hz}]$ and $z_s = 850\,[\mathrm{m}]$ with $f_0 = 8.0\,[\mathrm{Hz}]$). The resulting motions are scaled-up to a horizontal acceleration of 1 [g]. This acceleration level is consistent with the acceleration field due to a near field (within tens of kilometers) of a 45 °reverse-fault dipping source mechanism triggering a $M_w = 6.0$ event.

Figure 5.8 shows the displacement and accelerations recorded at the NPP site. Please note the presence of surface waves for the shallow-source case. This can be seen by noticing that after the maximum amplitude is reached, in-phase oscillatory motions remain. For the deep source case there is not much to be said in the ways of surface waves. This is expected in the light of the discussion in the preceding chapter. Also note that the shallow-source case has significant ($\approx 1\,[\mathrm{cm}]$) co-seismic displacement field in the $-X$ direction. The direction of this displacement coincides with that of the horizontal projection of the slip vector on the fault, which is a reverse fault, also in the $-X$ direction.

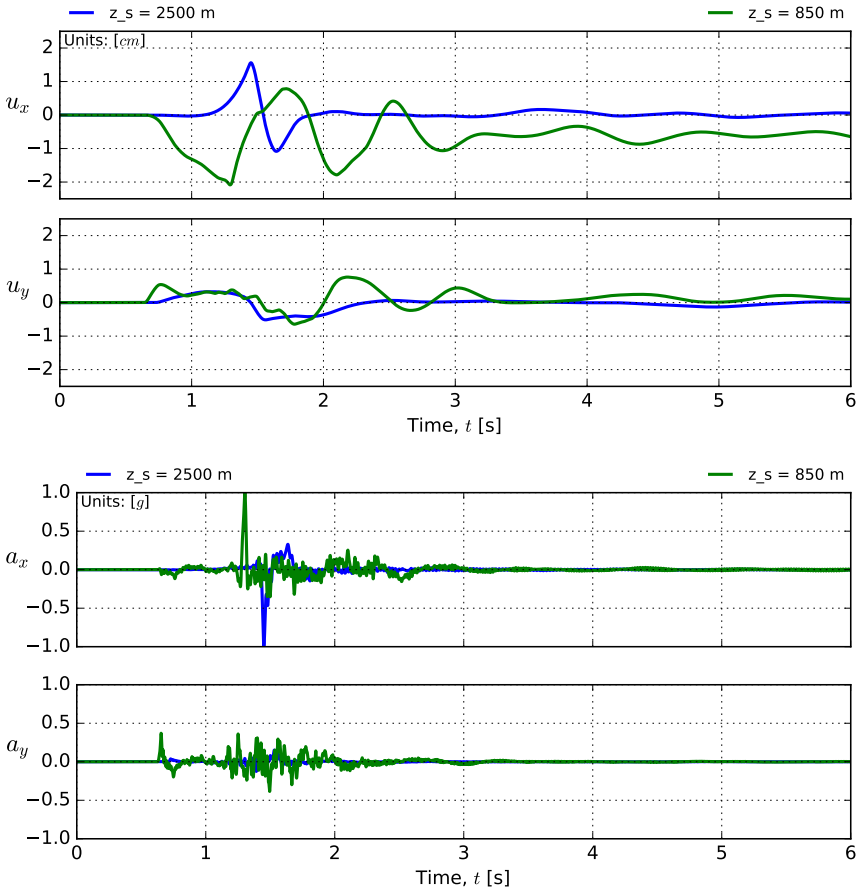


FIGURE 5.8. Displacement and acceleration histories for the generated earthquake motions at the NPP site.

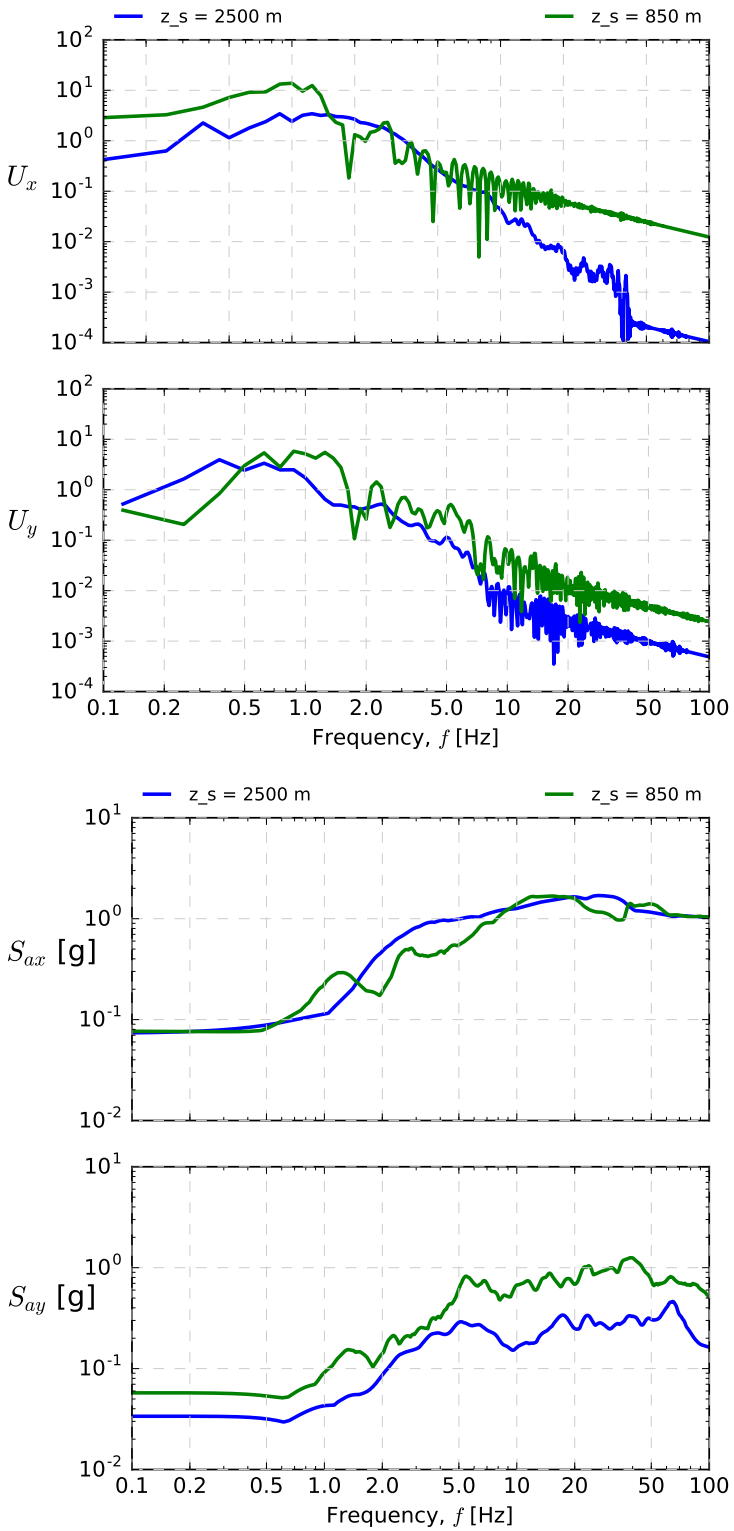


FIGURE 5.9. Fourier amplitude spectrum and 5%-damped acceleration response spectrum for the generated earthquake motions at the NPP site.

5.6. Results: Deep Source. Linear Model

The deep source at 45° dipping angle and corner frequency of $f_0 = 6.2\,[\text{Hz}]$ produces a seismic field which is approximatively unidimensional in the vicinity of the NPP site. Figure 5.10 shows the displacement response of the NPP measured at top of foundation and containment building. These results show high degree of agreement, and the small differences are no bigger than the ones obtained when using these motions to run the 1–D site analysis within ESSI to verify deconvolution.

The 1–D approximation is good in this case and this event will not be pursued further.

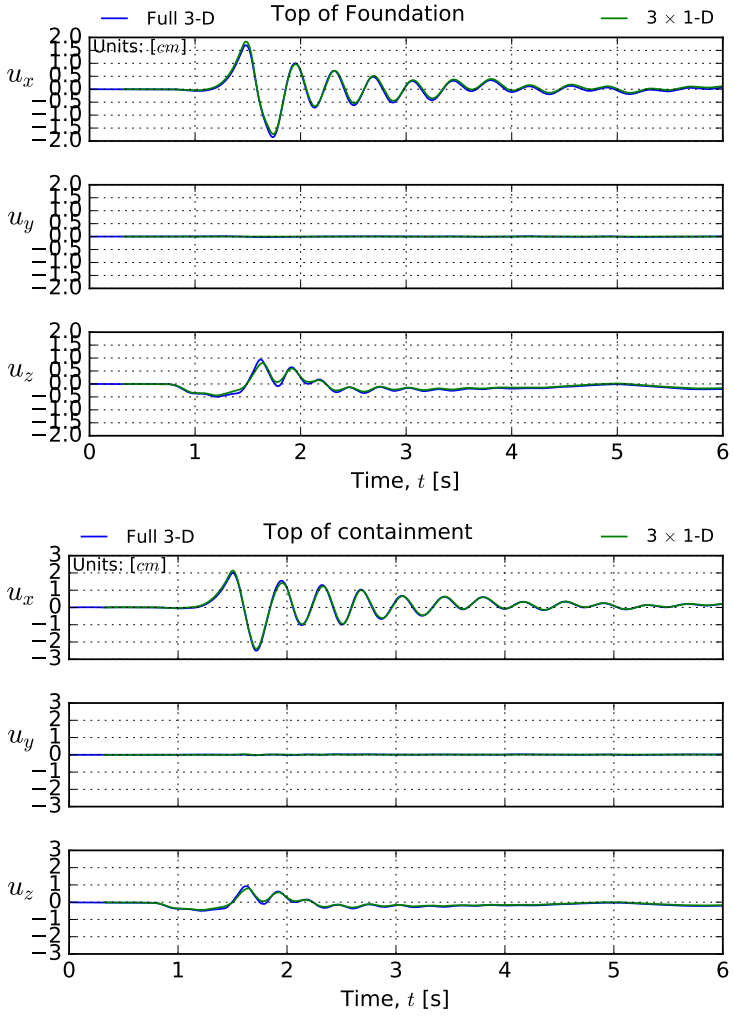
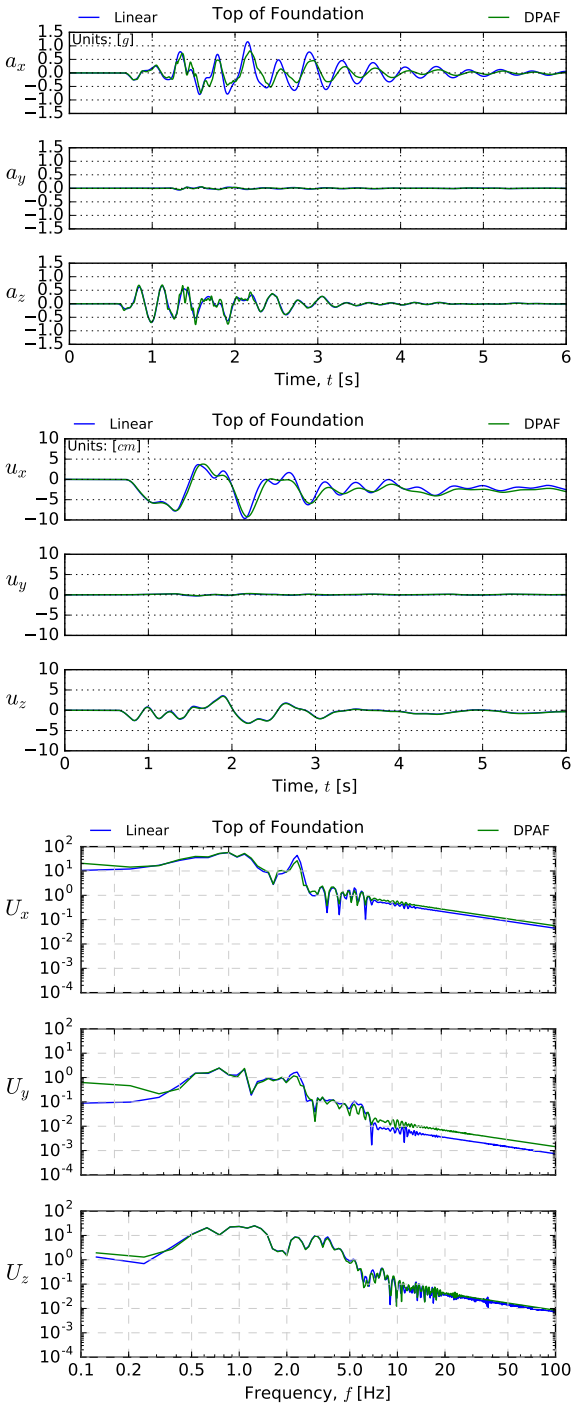


FIGURE 5.10. Displacement response histories at the top of foundation and containment building for seismic motions coming from a deeps source. Linear results.



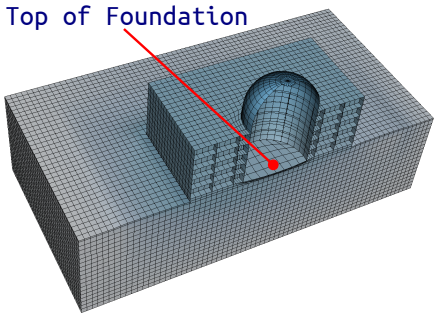
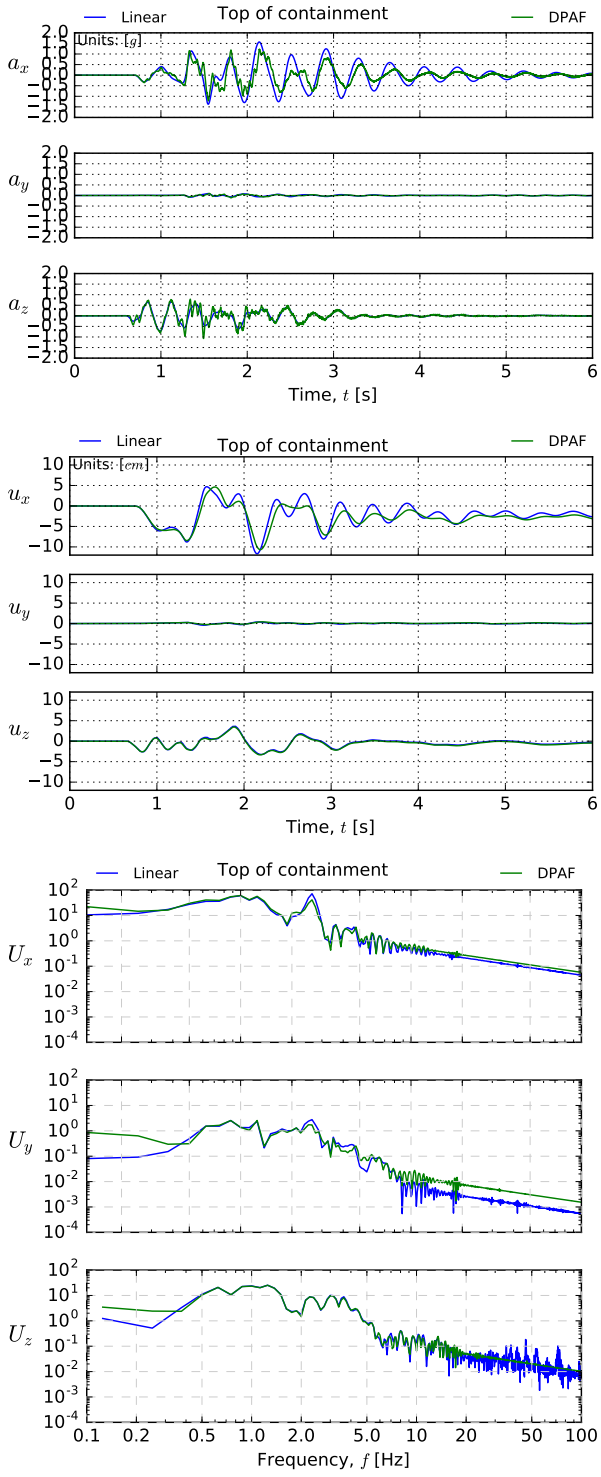


FIGURE 5.11. Structural response plots at the top of foundation comparing linear response to non-linear response, when subjected to 3–D motions.



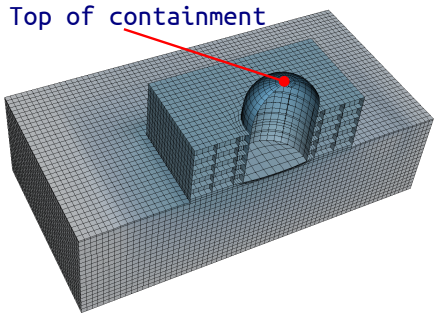
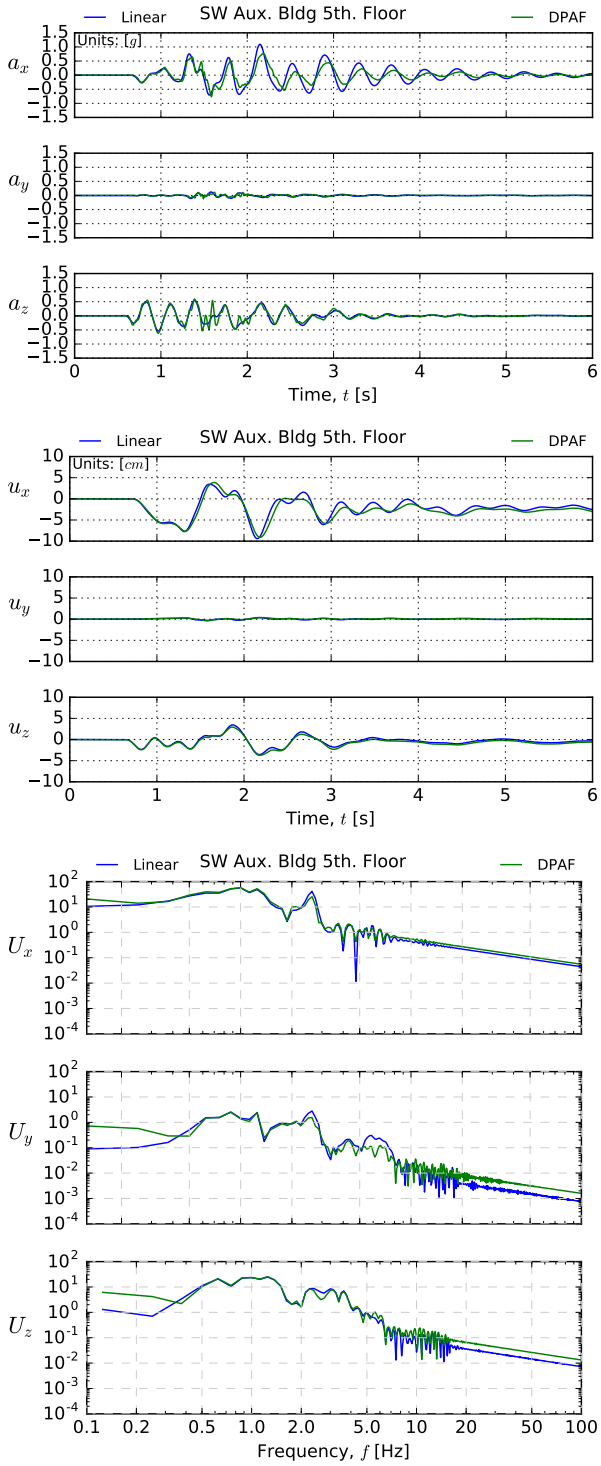


FIGURE 5.12. Structural response plots at the top of containment comparing linear response to non-linear one, when subjected to 3–D motions.



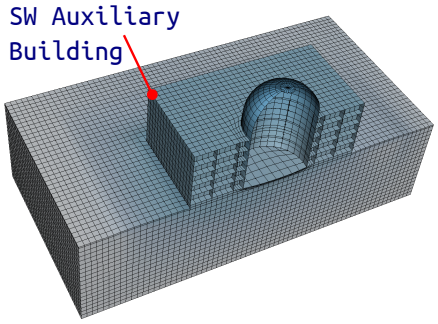
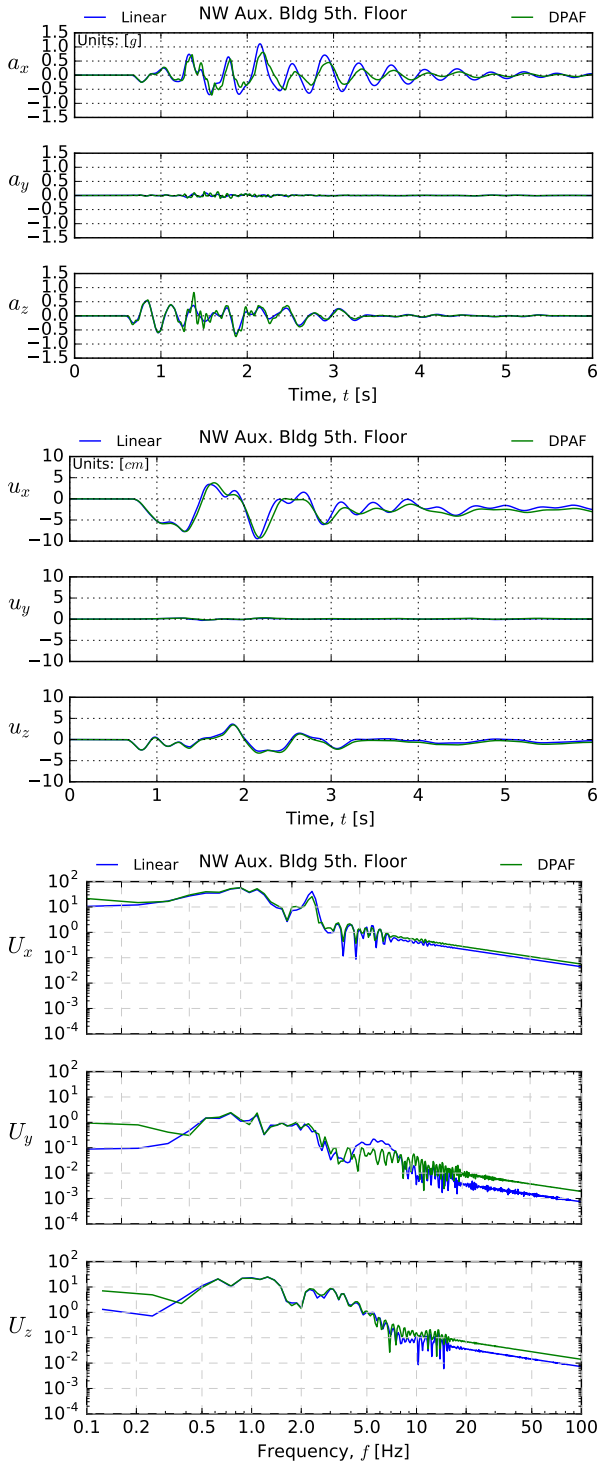


FIGURE 5.13. Structural response plots at the top south-west corner of auxiliary building comparing linear response to non-linear response, when subjected to 3–D motions.



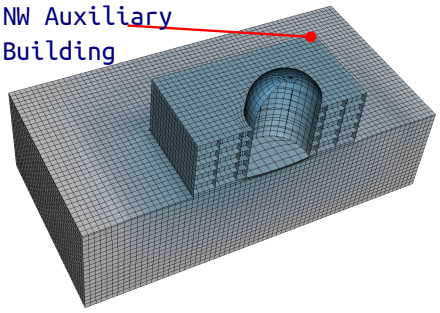


FIGURE 5.14. Structural response plots at the top north-west corner of auxiliary building comparing linear response to non-linear response, when subjected to 3–D motions.

5.8. Discussion of Results for Shallow Source, Linear vs. Non-Linear Models

With reference to Figures 5.11, 5.12, 5.13, and 5.14; linear response predicts accelerations slightly greater than the 1 [g] input motions throughout the auxiliary building plan and at the base of the foundation. Also, the top of the containment building responds in excess of 1.5 [g]. It is postulated that this increase in motion is due to rotations stemming from bending of the foundation slab and the considered flexibility of the foundation soil.

Figure 5.15 on page 116 shows the deformed shape of the NPP at peak displacement of the containment building, discarding the DRM layer, and also an exaggerated deformed shape of the foundation slab at this same time. Notice the deformation and rotation in the zone where the containment building is.

Reduction of peak horizontal accelerations was observed when non-linearity was considered. At the top of foundation, accelerations were reduced from 1.1 [g] to 0.8 [g]. Whereas at the top of containment building, accelerations were reduced from approximately 1.6 [g] to the neighborhood of 1.1 [g]. Similar results hold for horizontal response of auxiliary building. Conversely, vertical accelerations generally follow the linear response with the exception of fast transient spikes which, in some cases, increase the peak vertical acceleration response of the soil-structure system.

Peak displacement response are of approximately the same magnitude for both linear and non-linear cases throughout the NPP with the exception of the top of the containment building, which reduces it's maximum displacement from around 12 [cm] of total maximum displacement to about 10 [cm]. Nevertheless, visible in both acceleration and displacement histories are changes in the response shape and some evidence of slight period elongation. It is important to note that reported displacements and accelerations are absolute with respect to an inertial frame of reference. In addition to the observed static displacement field due to crustal deformation (fling), a permanent horizontal deformation in the order of 0.5 [cm] is observed throughout the plan. No permanent vertical settlements were observed.

Frequency response shows a consistent increase in low-frequency content throughout the plan. This is a manifestation of the elongation of the instantaneous vibration periods due to

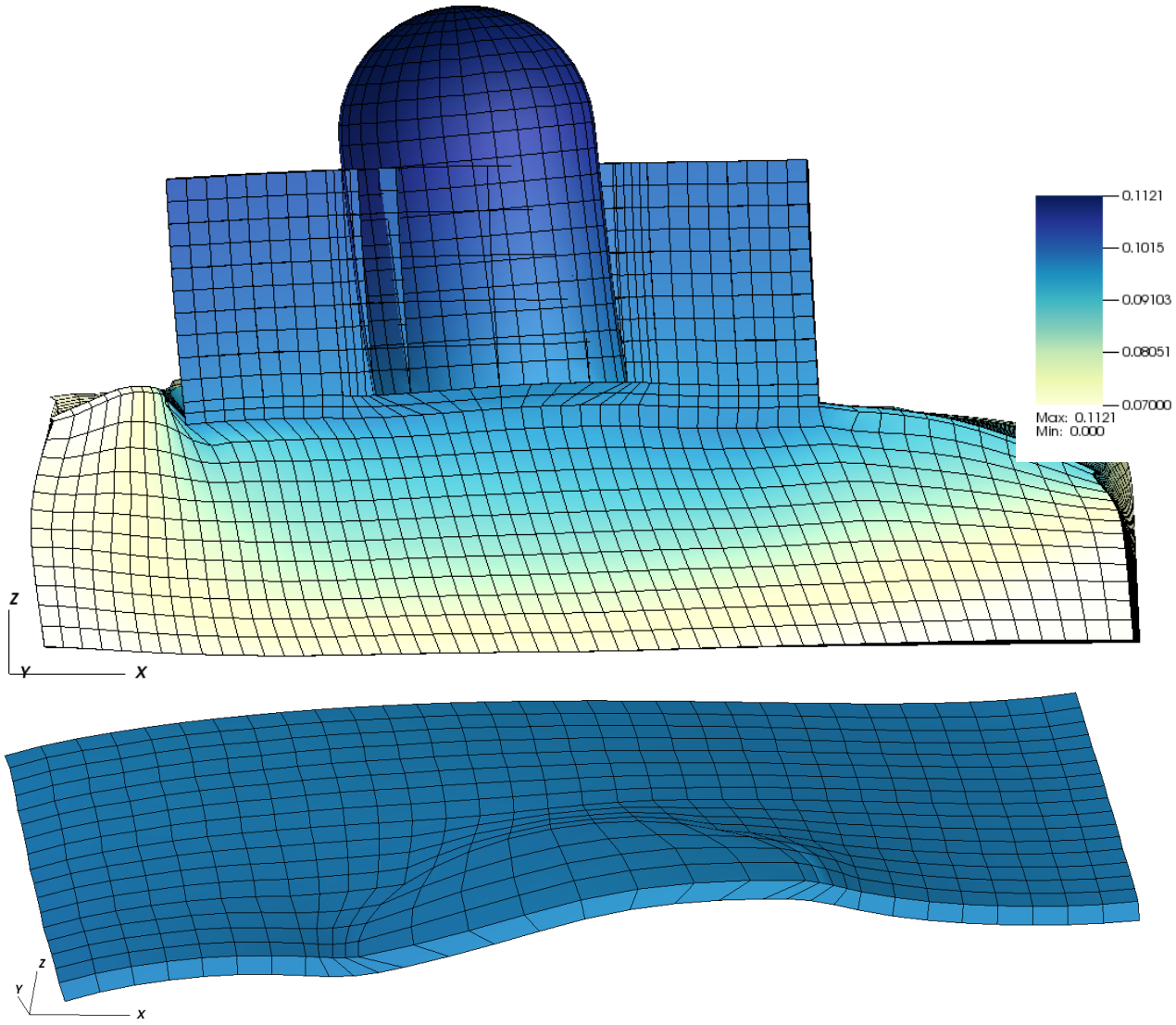


FIGURE 5.15. Deformed shape of the NPP and isolated foundation slab at peak displacement for the non-linear 3–D input case. Colorbar displacements in meters.

non-linearity. Not much can be said about the higher frequency ranges, there is some reduction of peak horizontal Fourier response throughout the plan, but otherwise not a significant modification of the high-frequency content (above 1 [Hz]).

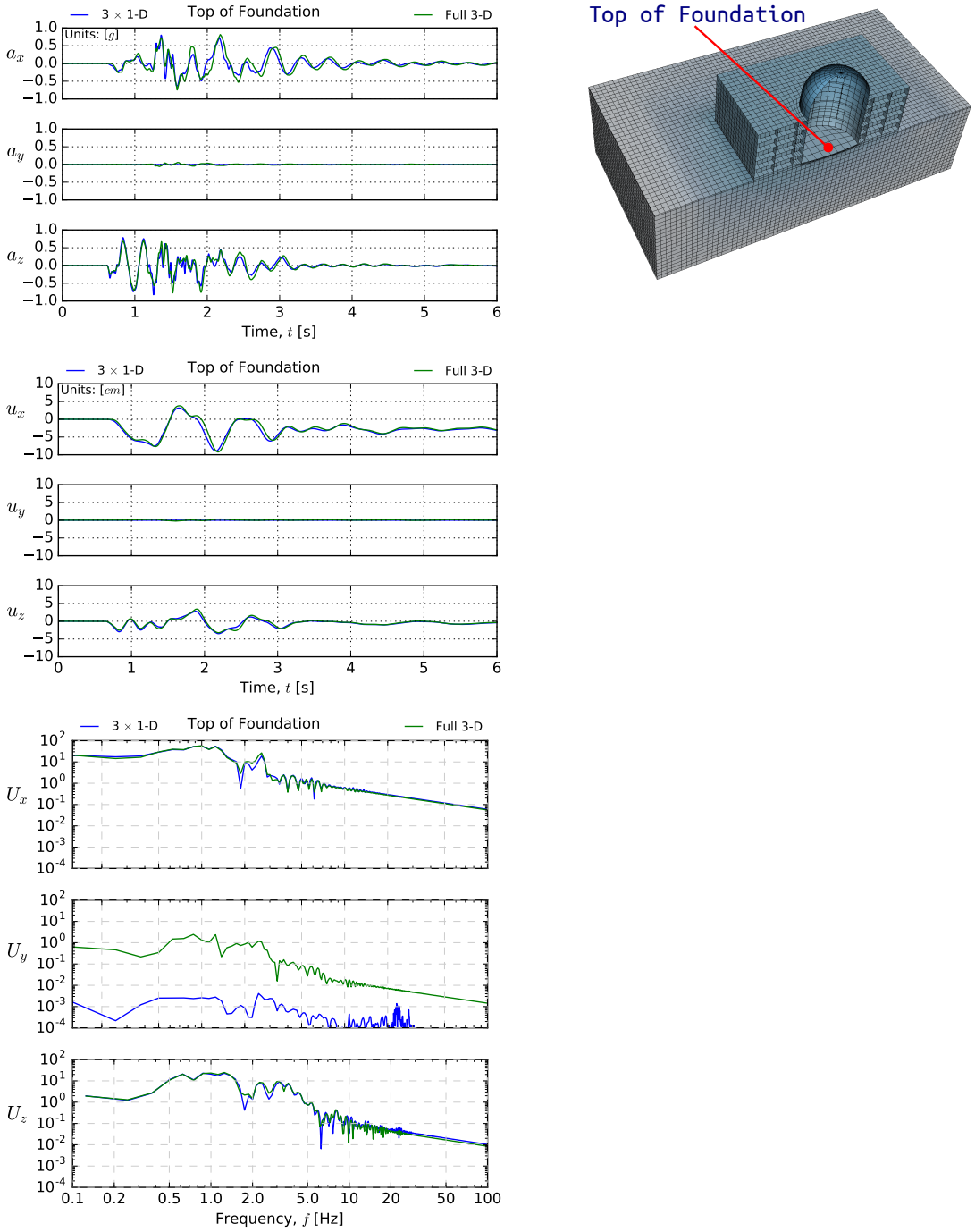
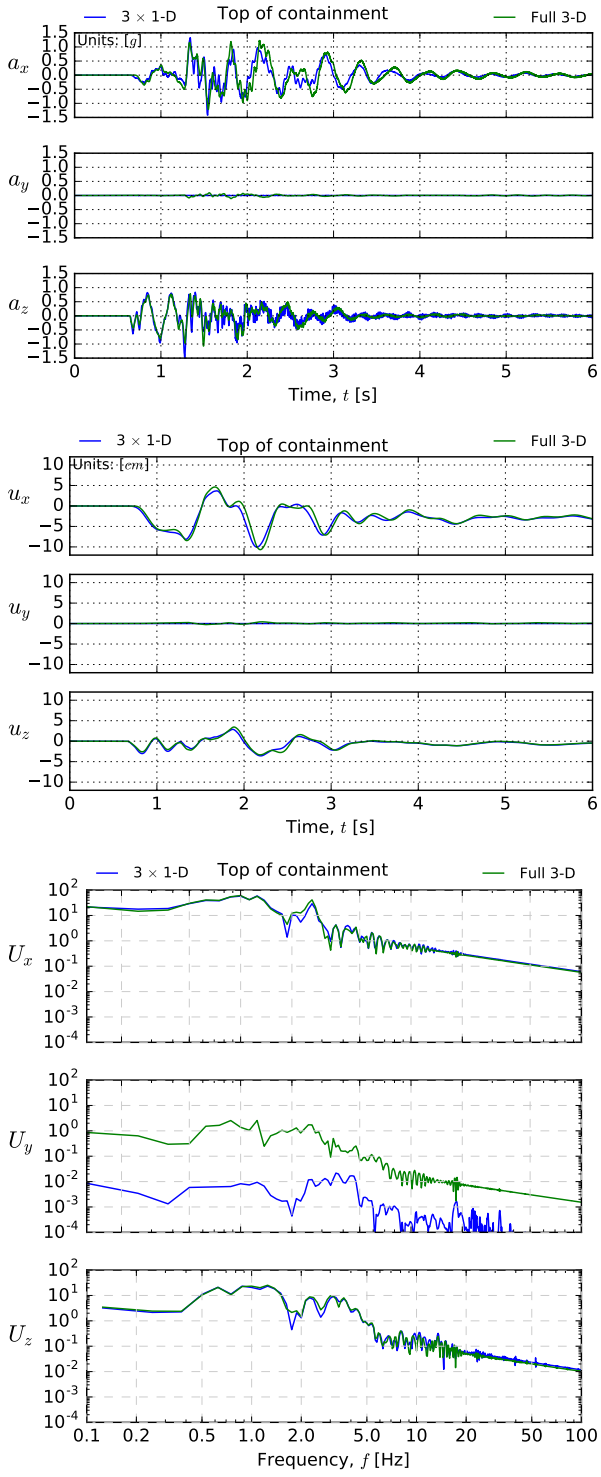


FIGURE 5.16. Structural response of the Non-linear sPP sub- jected to 3–D motions versucase when de-convolved 1–D, computed motions are used plots . The same non-linear model for soil was used.



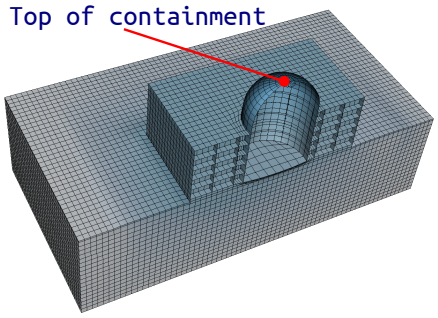
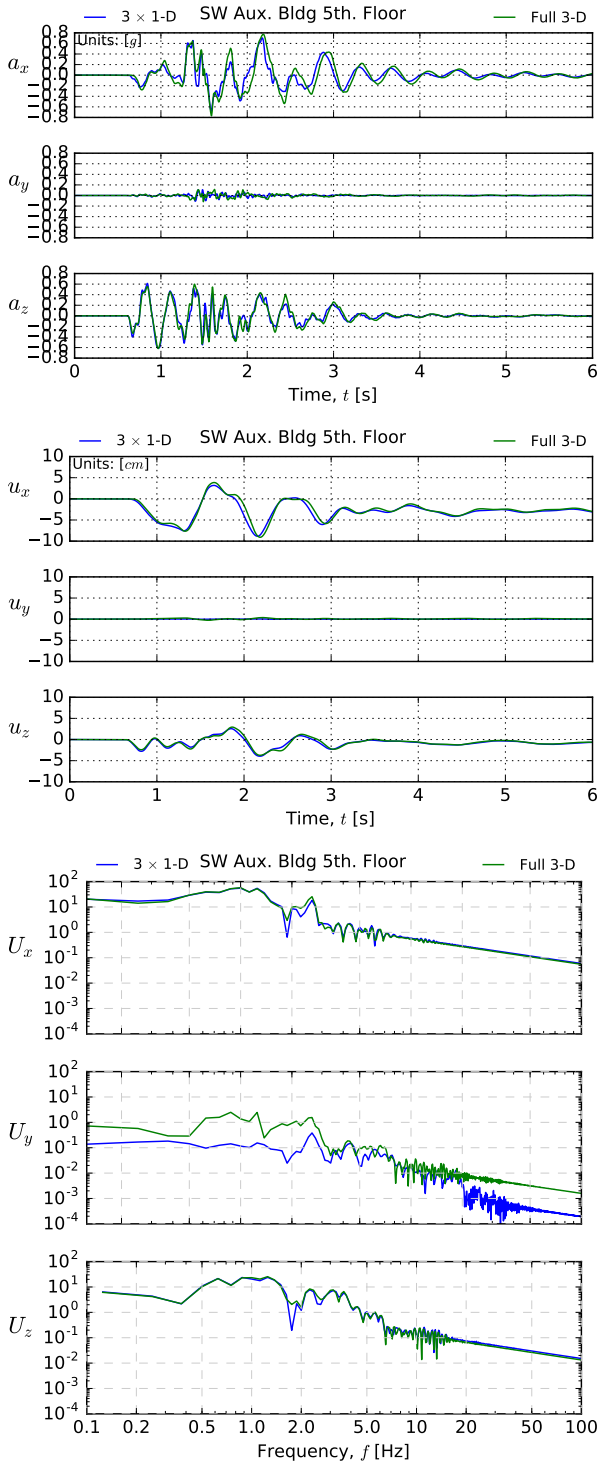


FIGURE 5.17. Non-linear structural response of the NPP subjected to 3–D motions versus the case when 1–D motions are used plots, computed at the top of containment comparing linear response .



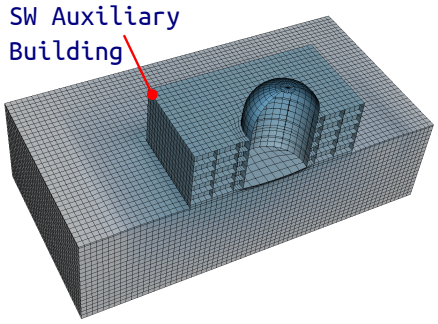
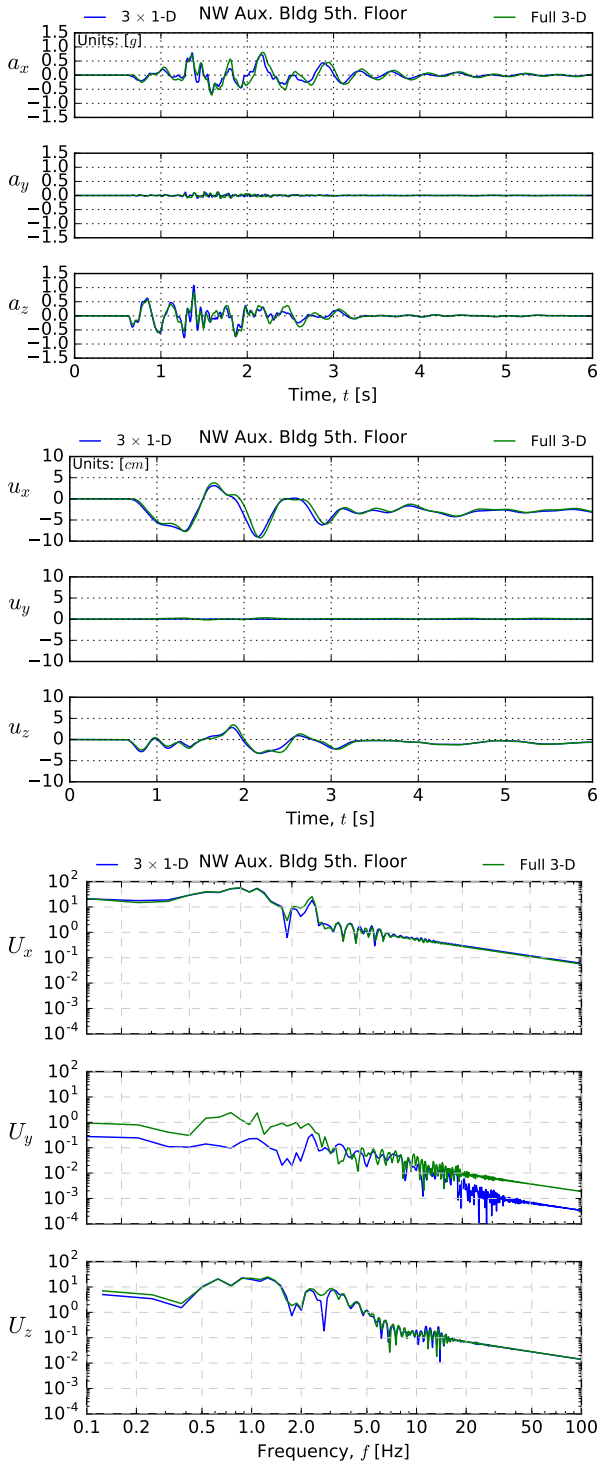


FIGURE 5.18. Non-linear structural response of the NPP subjected to 3–D motions versus the case when 1–D motions are used plots, computed at the top of south-west corner of auxiliary comparin.



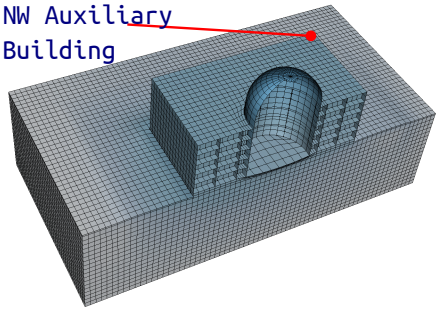


FIGURE 5.19. Non-linear structural response of the NPP subjected to 3–D motions versus the case when 1–D motions are used plots, computed at the top of north-west corner of auxiliary building.

5.10. Discussion of Results for Shallow Source, 3–D vs. 3×1–D Seismic Wave Fields

Recall, that the geologic situation for the shallow earthquake is that displayed in Figure 4.12. With reference to that figure, for the $z_s = 850\,[\mathrm{m}]$, for body-wave arrivals (prior to around the $t = 2.0\,[\mathrm{s}]$ mark) the site is behaving approximately in a 1–D fashion, with minor differences observed over a $500\,[\mathrm{m}]$ length-scale. The significant difference comes in the body-wave arrivals. Body waves are coming inclined into the site, but mostly from underneath it and with wavelengths larger of about site length for the frequencies considered. On the other hand, significant surface wave motions (albeit of less amplitude than those of the main body waves) roll into the site horizontally, differentially exciting the upper layers of the foundation site. 1–D site deconvolution will attribute horizontal and vertical motions which are due to surface waves to vertically propagating shear and compression waves, so it is not capable of capturing the correct directionality of the motion through the site for those arrivals.

This analysis explains the observed differences when comparing the two input fields in Figures 5.16, 5.17, 5.18, and 5.19; which occur mainly during the arrival of surface waves, after $t = 2\,[\mathrm{s}]$. Peak accelerations at these times are different, at many points of the model full 3–D analysis predicts higher peak response. There is no observable 'slab averaging' effect in this particular case.

Predicted residual horizontal deformation is virtually the same for both methods, as are the Fourier response spectrum which show minimal changes.

Based on these results it is expected that more drastic differences will arise for cases where the input motion is even more markedly three dimensional. Analysis at higher input frequencies are expected to yield more deviations from 1–D condition. Additionally, the 1–D method is expected to perform poorly in the near-field of a surface rupturing strike-slip event which would involve mostly horizontally propagating shear waves and Love waves. This has not been tested herein.

5.11. Energy Dissipation Distribution Results

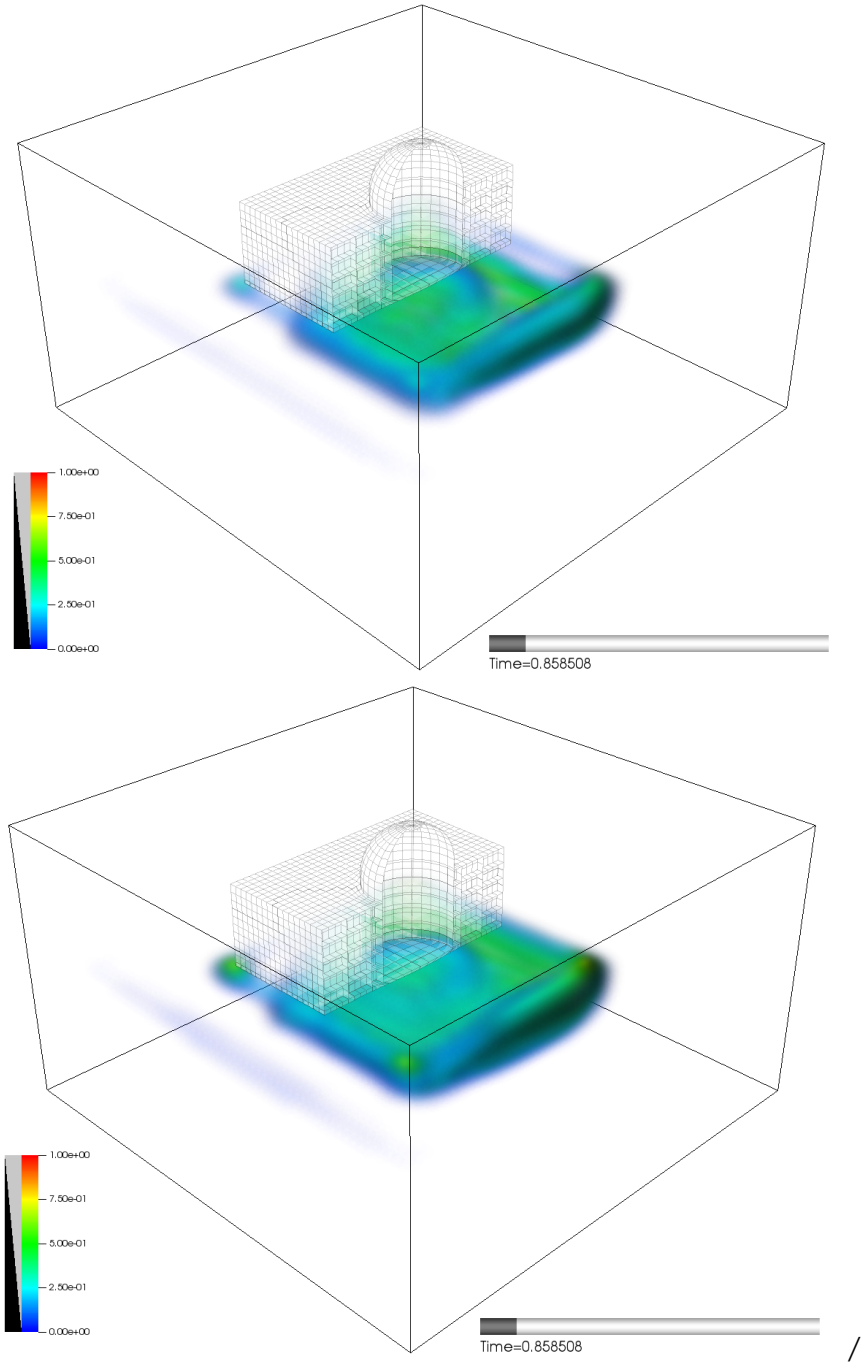


FIGURE 5.20. Volume render visualization of plastic energy dissipation rate in $\mathrm{kJ/s/m^3}$ at $t = 0.86\,\mathrm{sec}$ for soil under the NPP. **(Top)** result for full 3–D simulation. **(Bottom)** result for full 3×1–D simulation.

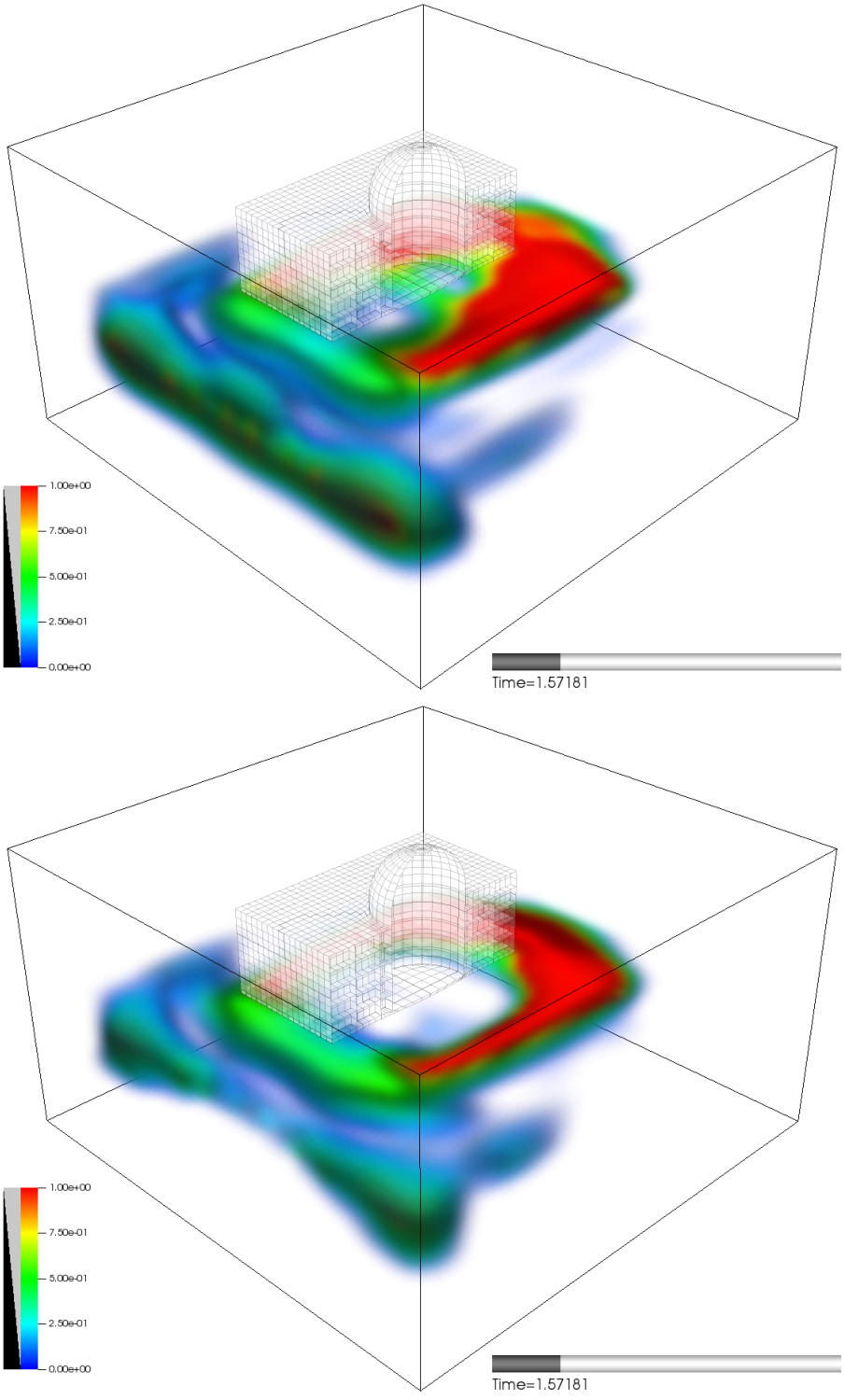


FIGURE 5.21. Volume render visualization of plastic energy dissipation rate in $\mathrm{kJ/s/m^3}$ at $t = 1.57\,\mathrm{sec}$ for soil under the NPP. **(Top)** result for full 3–D simulation. **(Bottom)** result for full 3×1–D simulation.

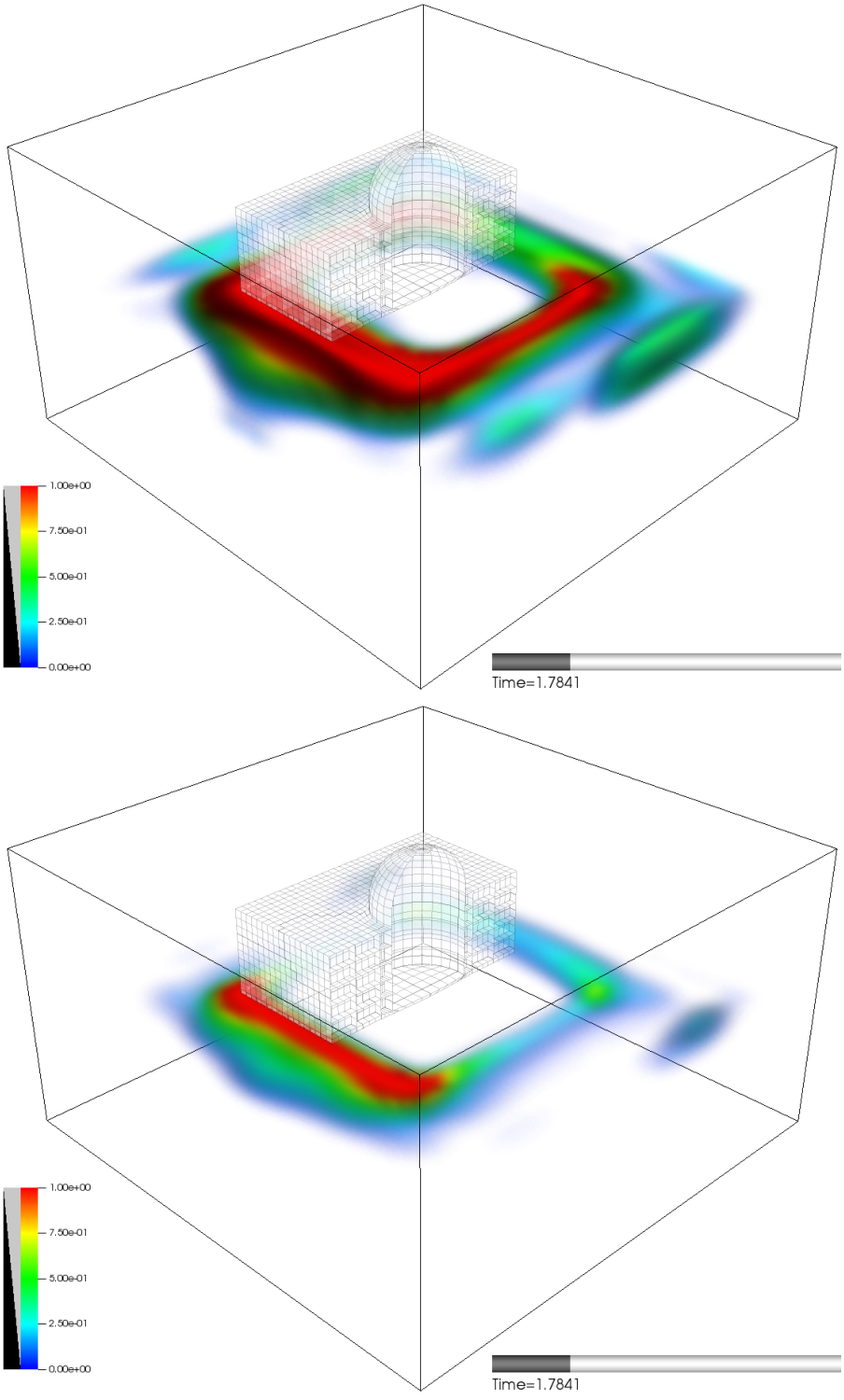


FIGURE 5.22. Volume render visualization of plastic energy dissipation rate in $\mathrm{kJ/s/m^3}$ at $t = 1.78\,\mathrm{sec}$ for soil under the NPP. **(Top)** result for full 3–D simulation. **(Bottom)** result for full 3×1–D simulation.

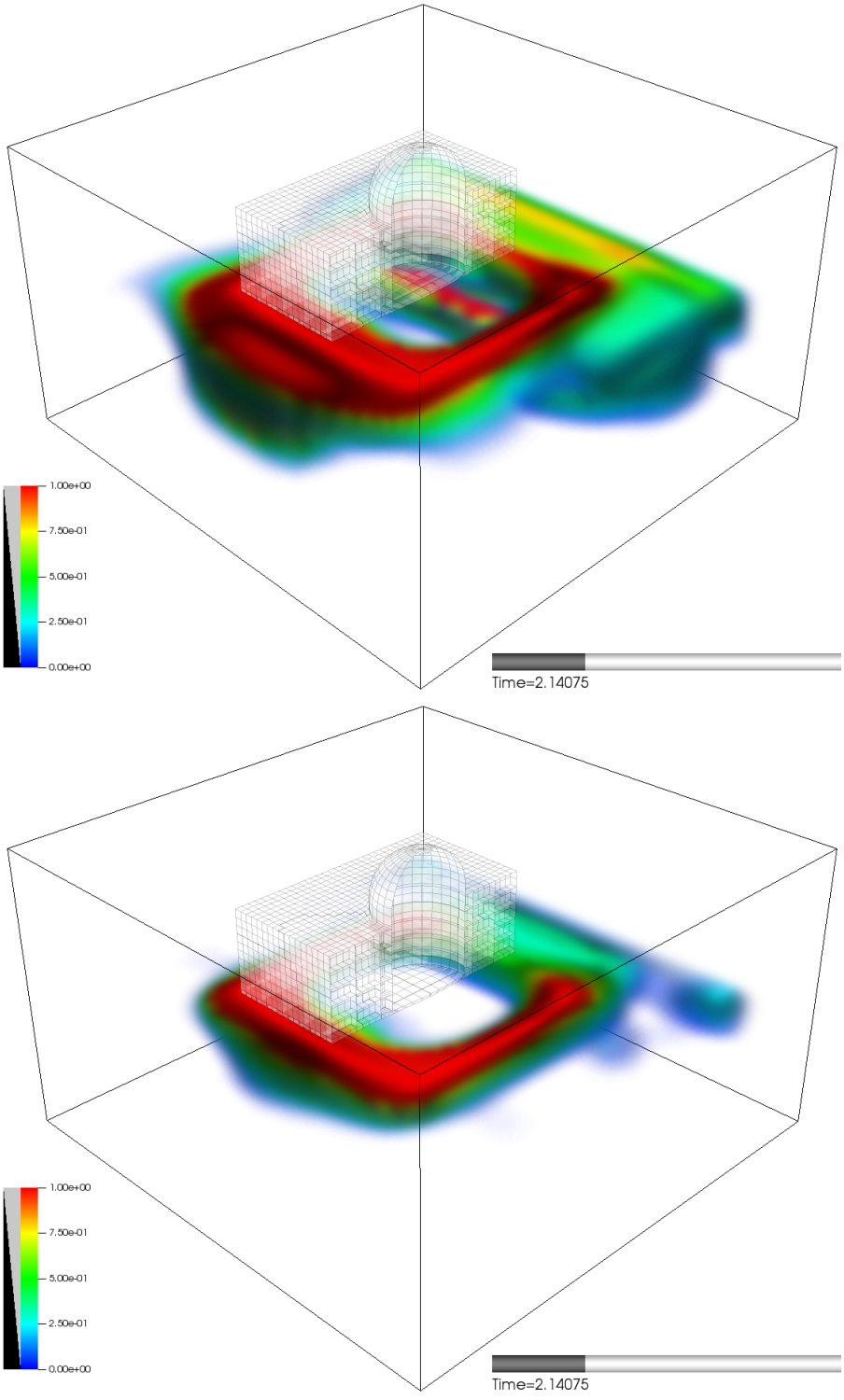


FIGURE 5.23. Volume render visualization of plastic energy dissipation rate in $\mathrm{kJ/s/m^3}$ at $t = 2.14\,\mathrm{sec}$ for soil under the NPP. **(Top)** result for full 3–D simulation. **(Bottom)** result for full 3×1–D simulation.

5.12. Discussion of Energy Dissipation Distribution Results

Figures 5.20, 5.21, 5.22, and 5.23 use a volume rendering technique to visualize the spatio-temporal distribution of energy dissipation rate within the non-linear soil, comparing full 3–D wave field to 3×1–D. Half of the NPP superstructure mesh is shown for reference as well as a bounding box which encompasses the complete FEM domain.

At peak response, volumes of soil are dissipating energy in excess of $3\,\mathrm{kJ/s/m^3}$.

It is interesting to note the consistent pattern that energy dissipation is occurring mostly around the perimeter of the foundation. In this area, vertical stresses due to NPP are large, with small horizontal stresses. This results in high deviatoric stress at somewhat low confinement, leading to high rates of plastic response. Towards the middle of the foundation, under the containment building, the vertical stresses are met with comparable horizontal stresses leading to higher confinement, requiring a larger deviatoric stress to produce plastic response.

Also notable is that the left side of the domain (south) is side closest to the earthquake source, and it is also there that plasticity is observed at the DRM/soil interface. Indeed, deformed shape plots show that there are plastic-deformation concentrations on this side of the domain. DRM motions occur on a linear domain, they quickly encounter non-linearity as they enter the soil volume. It is a yet unanswered question on how to deal with this situation properly, or if it is of some consequence to structural response. For now, it seems safe to assume that larger DRM domains might be needed so that this boundary effect (if significant at all) can be reduced. Of course, the best approach would be to place the DRM boundary at the rock-soil interface or, conceptually, where non-linearity is expected to be low. But this might not always be possible.

CHAPTER 6

Summary

Waves which emanate from a seismic source are three-dimensional from their outset, and get further modified by the propagation environment and their interaction with the free surface. For soil-structure interaction analysis, the validity of this assumption will depend on: the horizontal extension of the structure, the maximum considered frequency of seismic motions, and finally on the local geological setting and material properties. Especially in a perfectly layered medium, surface waves are present and may be an important contributor to overall motion, in direct contradiction to the 1–D site assumption. In a more heterogeneous environment there is no guarantee whatsoever that a particular site will experience a locally unidimensional wave field.

The response of a soil-structure system representing a typical nuclear power facility due to a near field earthquake, was computed and analyzed. Motions were calculated using a state-of-the-art parallel seismic simulation code. Without loss in generality, but only in realism, point sources were used to characterize scenario earthquakes. Indeed, point sources are the tool with which bigger earthquake models are realized. Therefore any features of the generated waves must stem only from these types of sources.

Fully 3–D DRM simulations of soil and structure agree with those obtained using a 3×1–D approach when the site response is approximately locally unidimensional. Upon perturbing the location of the source more surface waves were induced and propagated into the site. It was shown that it is not possible for a unidimensional model of wave propagation to correctly capture the effect these waves have. The reason is that surface waves will propagate horizontally into the site, inducing both vertical and horizontal components of ground motion. 3×1–D analysis will attribute these 'recorded' motions to horizontal waves and compression waves, with a degradation in accuracy of the computed response.

It is expected that these deviations from 1–D condition can only worsen with a heterogeneous environment, higher earthquake frequencies, and different source mechanisms. This makes full

3–D modeling of earthquake and structure indispensable when the structure of interest is of high importance.

Using the domain reduction method along with a correct model of earthquake source and local geology is a tool now available for practitioners to evaluate the response of structures to realistic earthquake scenarios. This is especially important for nuclear power plants which are very sensitive to soil-structure interaction and also require a great deal of confidence that their performance is adequate.

6.1. Future Lines of Research

The presented research can be expanded into different directions.

- Determination of the validity of the 1–D assumption in more extreme cases involving, perhaps, strike-slip earthquake sources which promote surface waves and also into higher frequencies.
- Computation of the response of structures to even more realistic earthquake models. Several finite-fault solutions to famous earthquake events have been provided and validated in the literature. These are readily available to be used as scenarios for new structures.
- Full validation of ESSI modeling using an existing instrumented structure, along with site information and a model of the earthquake to which this structure was subjected. This will provide the ultimate validation of the methodology.
- Inclusion and evaluation of the response differences when other parts of the system are modeled in greater detail. Examples include modeling foundation slipping and possible detachment using contact elements, evaluation of buoyancy effects in soil-embedded NPPs with high water-tables, modeling the internal structure and other equipment explicitly, simulating base-isolated structures, to name a few.
- Investigating the role of soil-dilation in the response of structures subjected to 3–D loading.
- Characterizing the properties of 3–D wave propagation in non-linear medium using this methodology is possible. This will prove important to reduce the uncertainty in site

response computations when non-linearity is important. In this same line, acceleration records in sites exhibiting non-linearity can be predicted and compared if a model of the earthquake is available. This would provide more evidence for the success of this approach to modeling.

Finally, transferring these tools into practice for routine use will require deep research into efficient implementations of the involved algorithms at all scales. The role of high-performance computing in this endeavor cannot be overstated. It is of great importance that verified and validated, high-performance, easy to use tools for modeling of ESSI problems become available to every day practice. It is also important that the terms of usage of the tools (licenses) are adequate to allow scaling in size of the problem without incurring in excessive costs apart from computational time.

This is the only way that practitioners will see the benefits of these methods an put them to good use. This, along with a conscious and honest-to-truth effort on part of authorities to regulate the means and methods by which we evaluate the safety of our infrastructure will no-doubt bring greater seismic performance to our built environment and peace to our minds.

APPENDIX A

Review of the Non-Linear Finite-Element Method for Small-Deformations

A.1. Finite-element method for small deformation dynamic problems with material non-linearity

For completeness, this section will cover the key concepts and terminology relating to non-linear finite element method (FEM). The reader is advised to refer to the excellent literature on the Finite Element Method for a deeper treatment of the subject. An outstanding example, followed somewhat in this text, is 'The Finite Element Method' by Hughes (2000).

The Finite Element Method (FEM) is an established solution scheme for a wide range of partial differential equations given their boundary conditions and initial field. The linear theory of FEM is well established, while the non-linear version is less so. In practical problems with physical sense, the non-linear solution is usually achieved by applying the solution scheme incrementally through a series of ordered time-steps, tracing the causality path that leads to a particular problem configuration under the assumption that that the problem is linear between time-steps.

Attention will be focused on the case of dynamic elasto-plasticity, which is a particular family partial differential equations. Figure A.1 shows an abstract depiction of an elasto-plastic domain where the dynamic elasto-plastic equations of motion are to be solved. The vector field $\boldsymbol{u}(\boldsymbol{x}, t)$ defined for $x \in \Omega \subset \Re^3$ is called the displacement field and is the solution to the problem. The boundary of the problem $\Gamma = \partial\Omega$ is divided into an essential boundary, $\Gamma_{\text{essential}}$, and a natural one, Γ_{natural}. On the essential boundary are defined boundary conditions which set the value of the displacement field directly, while at the natural boundary a traction $\boldsymbol{t} = \boldsymbol{\sigma} \cdot \boldsymbol{n}$ is defined. $\boldsymbol{\sigma}$ is the Cauchy stress tensor, while $\boldsymbol{n}$ is the outward normal to the domain. Not depicted is a body force-field $\boldsymbol{F}(\boldsymbol{x}, t)$ defined at each point in the domain.

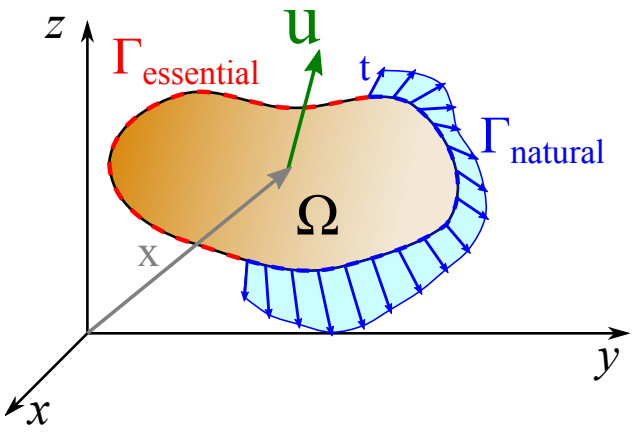


FIGURE A.1. Illustration of the domain within which the dynamic elasto-plastic equations of motion are to be solved.

The boundary conditions can be written mathematically in terms of their Cartesian tensor components:

$$u_i = 0 \qquad \boldsymbol{x} \in \Gamma_{\text{natural}} \tag{A.1}$$

$$\sigma_{ij} n_j = t_i \qquad \boldsymbol{x} \in \Gamma_{\text{essential}} \tag{A.2}$$

$$u_i^0 = u_i^0(\boldsymbol{x}) \qquad \boldsymbol{x} \in \Omega \tag{A.3}$$

Where the Einstein (Einstein, 1916) summation convention is implied on all repeated indexes. Equation A.1 shows the essential boundary condition, though it is not necessary that this be a homogeneous condition it is assumed for expositional simplicity. Equation A.1 shows how the traction boundary condition is represented as a natural boundary condition. Finally, Equation A.3 shows a function defining the initial displacement field for the problem.

Additionally, it will be assumed that it suffices for this exposition to model material strains using small-deformation theory. In this theory the strain-tensor, energy conjugate of the Cauchy stress tensor, is the symmetrized deformation gradient.

$$\epsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right) \tag{A.4}$$

FEM equations are obtained by applying the principle of virtual work to establish (static or dynamic) equilibrium conditions for the domain. Suppose a variation δu_i of the displacement

field such that it satisfies Equation A.1, the essential boundary conditions, the virtual work principle states that the if the system is in equilibrium then the virtual work done by internal forces δW_{int} must equal that done by the external forces δW_{ext}. The internal work done by the virtually straining the stress tensor can be shown to be:

$$\delta W_{\text{int}} = \int_V \sigma_{ij} \delta \epsilon_{ij} \, \mathrm{d}V \tag{A.5}$$

Where $\delta \epsilon_{ij}$ is the virtual small-deformation strain tensor produced by the virtual displacement field δu_i. The external work is done by the body forces and the surface forces. Body forces include the field $\boldsymbol{F}(\boldsymbol{x}, t)$ as well as the field of inertial forces $-\rho \ddot{\boldsymbol{u}}(\boldsymbol{x}, t)$, when displaced though the virtual displacement field $\delta \boldsymbol{u}(\boldsymbol{x})$. The virtual work done by external forces is, then

$$\delta W_{\text{ext}} = -\int_V \rho \ddot{u}_i \delta u_i \, \mathrm{d}V + \int_V F_i \delta u_i \, \mathrm{d}V + \int_{\Gamma_{\text{essential}}} t_i \delta u_i \, \mathrm{d}S$$

Then, applying the principle of virtual work:

$$\delta W_{\text{int}} = \delta W_{\text{ext}} \tag{A.6}$$

$$\int_V \sigma_{ij} \delta \epsilon_{ij} \, \mathrm{d}V = -\int_V \rho \ddot{u}_i \delta u_i \, \mathrm{d}V + \int_V F_i \delta u_i \, \mathrm{d}V + \int_{\Gamma_{\text{essential}}} t_i \delta u_i \, \mathrm{d}S \tag{A.7}$$

$$\int_V \rho \ddot{u}_i \delta u_i \, \mathrm{d}V + \int_V \sigma_{ij} \delta \epsilon_{ij} \, \mathrm{d}V = \int_V F_i \delta u_i \, \mathrm{d}V + \int_{\Gamma_{\text{essential}}} t_i \delta u_i \, \mathrm{d}S \tag{A.8}$$

Equation A.8 is known as the 'weak-form' or variational formulation of the partial differential equation of motion. Solutions $\boldsymbol{u}$ must satisfy Equation A.8 for any variation (or virtual displacement field) $\delta \boldsymbol{u}$ that satisfies the essential boundary conditions. This result can also be obtained using variational calculus or also starting from the dynamic partial differential equation of motion, multiplying with a test function, integrating over the volume and then invoking integration by parts to bring out boundary terms and identifying each term as above.

The finite element method then proceeds to define a discretization for the displacement field, that is, a basis of interpolation functions which define a mesh of elements over the domain. Although it is possible to consider a mesh for the complete domain in this process, it is most beneficial to develop element-by-element equilibrium conditions which are then *assembled* into an approximation of the domain. To illustrate the process, consider an abstract element with N_e

nodes or spatial points at which the displacements are known, the displacement field inside the element $\boldsymbol{u}_e(\boldsymbol{x})$ is interpolated from nodal displacement using appropriate interpolation functions basis. That is,

$$u_i(x_1, x_2, x_3) = N_{ijp}(x_1, x_2, x_3)u_{jp} \qquad i, j = \{1, 2, 3\} \text{ and } p = \{1, 2, \ldots, N_e\} \tag{A.9}$$

Where u_{jp} is the j-th Cartesian component of the nodal displacement at element node number p. $N_{ijp}(x_1, x_2, x_3)$ are functions that interpolate the nodal displacements at any point inside the element.

From this, the interpolated strain field is obtained by replacing Equation A.9 into equation A.4

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial}{\partial x_j}\left[N_{ikp}(x_1, x_2, x_3)u_{kp}\right] + \frac{\partial}{\partial x_i}\left[N_{jkp}(x_1, x_2, x_3)u_{kp}\right]\right) \tag{A.10}$$

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial N_{ikp}}{\partial x_j} + \frac{\partial N_{jkp}}{\partial x_i}\right)u_{kp} \tag{A.11}$$

It is convenient to have a reference geometry for elements on which the integrals involved in Equation A.8 can be evaluated easily. To this end define a set of *element-local coordinates*, or simply 'local coordinates', $\{\xi_i\}_{i=1}^3$ such that $\xi_i \in [-1, 1]$, $\forall i$. For example, these coordinates might span a unit cube for 3–D elements, or a unit square for plane elements, and the unit line segment for line elements, and so on. Interpolation functions written in this reference coordinate system would only have to be written once, whereas in a general system they would need to be developed independently for each element. So now the interpolation functions are functions of the local coordinates, $N_{ikp}(\xi_1, \xi_2, \xi_3)$, and there exists a unique transformation from the reference element to the element in global coordinate system. If the same interpolation functions used to define the geometry as are used to also describe the displacement field and its variations, the element is said to be *isoparametric* and the local to global coordinate transformation for an element will be:

$$x_i(\xi_1, \xi_2, \xi_3) = N_{ikp}(\xi_1, \xi_2, \xi_3)X_{kp} \tag{A.12}$$

Where X_{kp} is the k-th coordinate of the p-th node of the element. Then, the strain tensor, given by Equation A.11, must be evaluated using the chain rule of differentiation.

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial N_{ikp}}{\partial x_j} + \frac{\partial N_{jkp}}{\partial x_i}\right) u_{kp} \tag{A.13}$$

$$= \underbrace{\frac{1}{2}\left(\frac{\partial N_{ikp}}{\partial \xi_a}\frac{\partial \xi_a}{\partial x_j} + \frac{\partial N_{jkp}}{\partial \xi_b}\frac{\partial \xi_b}{\partial x_i}\right)}_{B_{ikpj}} u_{kp} \tag{A.14}$$

$$= B_{ikpj} u_{kp} \tag{A.15}$$

B_{ikpj} relates strains within the element to element nodal displacements, through the interpolation functions, and is called the 'strain-displacement tensor'.

Noting that variations of the interpolated displacement field can be obtained by applying variations to the nodal displacement values, Equation A.8 can be written in interpolated form for an element of volume V_e.

$$\int_{V_e} \rho \ddot{u}_i \delta u_i \, \mathrm{d}V_e + \int_{V_e} \sigma_{ij} \delta \epsilon_{ij} \, \mathrm{d}V_e = \int_{V_e} F_i \delta u_i \, \mathrm{d}V_e \tag{A.16}$$

$$\int_{V_e} \rho N_{ilq} \ddot{u}_{lq} N_{ikp} \delta u_{kp} \, \mathrm{d}V_e + \int_{V_e} \sigma_{ij} B_{ikpj} \delta u_{kp} \, \mathrm{d}V_e = \int_{V_e} F_i N_{ikp} \delta u_{kp} \, \mathrm{d}V_e \tag{A.17}$$

$$l = \{1, 2, 3\} \text{ and } q = \{1, 2, \ldots, N_e\}$$

$$\left\{\int_{V_e} \rho N_{ilq} \ddot{u}_{lq} N_{ikp} \, \mathrm{d}V_e + \int_{V_e} \sigma_{ij} B_{ikpj} \, \mathrm{d}V_e\right\} \delta u_{kp} = \left\{\int_{V_e} F_i N_{ikp} \, \mathrm{d}V_e\right\} \delta u_{kp} \tag{A.18}$$

By virtue of the arbitrary nature of the variation of the nodal displacements δu_{kp}, the virtual nodal displacements, δu_{kp}, an be eliminated of Equation A.18.

$$\left(\int_{V_e} \rho N_{ilq} N_{ikp} \, \mathrm{d}V_e\right) \ddot{u}_{lq} + \int_{V_e} \sigma_{ij} B_{ikpj} \, \mathrm{d}V_e = \int_{V_e} F_i N_{ikp} \, \mathrm{d}V_e \tag{A.19}$$

If the stress-strain relationship of the material is non-linear, then Equation A.19 can be evaluated incrementally. Applying force increments ΔF_i, which add up to F_i, incremental nodal displacements Δu_{lq} and accelerations $\Delta \ddot{u}_{lq}$ can be computed. The stress-strain relationship is also evaluated incrementally via the constitutive rate equations

$$\Delta \sigma_{ij} = E^t_{ijkl} \Delta \epsilon_{kl} \tag{A.20}$$

Such that $\sigma_{ij} = \sum \Delta\sigma_{ij}$ with the sum spanning all force increments. E^t_{ijkl} is a tangent elasticity tensor, which will depend on the loading history for each material point. The virtual work equation, Equation A.19 must be satisfied for each increment, otherwise equilibrium would be lost after the step. Therefore:

$$\left(\int_{V_e} \rho N_{ilq} N_{ikp} \, \mathrm{d}V_e\right) \Delta\ddot{u}_{lq} + \int_{V_e} \Delta\sigma_{ij} B_{ikpj} \, \mathrm{d}V_e = \int_{V_e} \Delta F_i N_{ikp} \, \mathrm{d}V_e \tag{A.21}$$

Using Equation A.20 on the equation above

$$\left(\int_{V_e} \rho N_{ilq} N_{ikp} \, \mathrm{d}V_e\right) \Delta\ddot{u}_{lq} + \int_{V_e} E^t_{ijmn} \Delta\epsilon_{mn} B_{ikpj} \, \mathrm{d}V_e = \int_{V_e} \Delta F_i N_{ikp} \, \mathrm{d}V_e \tag{A.22}$$

Using Equation A.15, the increment in strain can be computed from the increment in nodal displacements. This is replaced into the equation above to yield

$$\left(\int_{V_e} \rho N_{ilq} N_{ikp} \, \mathrm{d}V_e\right) \Delta\ddot{u}_{lq} + \int_{V_e} E^t_{ijmn} B_{mlqn} \Delta u_{lq} B_{ikpj} \, \mathrm{d}V_e = \int_{V_e} \Delta F_i N_{ikp} \, \mathrm{d}V_e \tag{A.23}$$

$$\left(\int_{V_e} \rho N_{ilq} N_{ikp} \, \mathrm{d}V_e\right) \Delta\ddot{u}_{lq} + \left(\int_{V_e} E^t_{ijmn} B_{mlqn} B_{ikpj} \, \mathrm{d}V_e\right) \Delta u_{lq} = \int_{V_e} \Delta F_i N_{ikp} \, \mathrm{d}V_e \tag{A.24}$$

$$M_{kplq} \Delta\ddot{u}_{lq} + K^t_{kplq} \Delta u_{lq} = \Delta\tilde{F}_{kp} \tag{A.25}$$

Where K^t_{kplq} is the tangent stiffness tensor for the element, and M_{kplq} is the mass tensor for the element, and $\Delta\tilde{F}_{kp}$ is the nodal-equivalent body force tensor. It is usual to unroll u_{lq} into a vector $\Delta\boldsymbol{u}$ of incremental nodal displacements. In this case $\boldsymbol{K}^t_e$ and $\boldsymbol{M}_e$ become tangent stiffness and mass matrices respectively and $\Delta\tilde{\boldsymbol{F}}$ is the nodal-equivalent body force vector.

$$\boldsymbol{M}_e \Delta\ddot{\boldsymbol{u}}_e + \boldsymbol{K}^t_e \Delta\boldsymbol{u}_e = \Delta\tilde{\boldsymbol{F}}_e \tag{A.26}$$

Now in order to obtain these matrices, the integrals that define them have to be evaluated. This is usually done via Gaussian quadrature on the reference domain (local coordinates). The stiffnes tensor term is:

$$K^t_{kplq} = \int_{V_e} B_{ikpj} E^t_{ijmn} B_{mlqn} \, \mathrm{d}V_e \tag{A.27}$$

First, the integral is re-stated in terms of the local coordinate system $\{\xi_i\}_{i=1}^{3}$,

$$K^t_{kplq} = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} B_{ikpj} E^t_{ijmn} B_{mlqn} |J| \, \mathrm{d}\xi_1 \, \mathrm{d}\xi_2 \, \mathrm{d}\xi_3 \tag{A.28}$$

Where $|J|$ is the determinant of the Jacobian of the coordinate transformation relating the reference domain and the current element, defined by Equation A.12. Then, a set of N_g 'Gauss-Points', $\left\{\boldsymbol{\xi}_g\right\}_{g=1}^{N_g}$, with associated integration weights, $\left\{w_g\right\}_{g=1}^{N_g}$, are defined in the reference domain at which points the involved terms of the strain-displacement tensors, strains and tangent elastic moduli are evaluated. The integral in Equation A.28 is approximated then by

$$K^t_{kplq} = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} B_{ikpj} E^t_{ijmn} B_{mlqn} |J| \, \mathrm{d}\xi_1 \, \mathrm{d}\xi_2 \, \mathrm{d}\xi_3 \tag{A.29}$$

$$\approx \sum_{g=1}^{N_g} w_g B_{ikpj}(\boldsymbol{\xi}_g) E^t_{ijmn}(\boldsymbol{\xi}_g) B_{mlqn}(\boldsymbol{\xi}_g) |J(\boldsymbol{\xi}_g)| \tag{A.30}$$

After evaluating the involved matrices for each element in the mesh, the global equation of motion can be assembled by adding up contributions from elements into the corresponding rows of the global matrix. This leads to a global non-linear set of second order differential equations which represent a discretization of the equations of motion of the system.

$$\boldsymbol{M}\Delta\ddot{\boldsymbol{u}} + \boldsymbol{K}^t \Delta\boldsymbol{u} = \Delta\tilde{\boldsymbol{F}} \tag{A.31}$$

Where now $\boldsymbol{M}$ is a global mass matrix and $\boldsymbol{K}^t$ is a global tangent stiffness equation, while $\Delta\boldsymbol{u}$ represent increment of global nodal displacements and $\Delta\tilde{\boldsymbol{F}}$ global increments in equivalent body forces.

A damping matrix $\boldsymbol{C}$ may be added to complete the formulation

$$\boldsymbol{M}\Delta\ddot{\boldsymbol{u}} + \boldsymbol{C}\Delta\dot{\boldsymbol{u}} + \boldsymbol{K}^t \Delta\boldsymbol{u} = \Delta\tilde{\boldsymbol{F}} \tag{A.32}$$

This matrix may come from the underlying physics, e.g. for coupled porous-material and fluid element formulations or for actual physical damping devices, or might be added based on empirical observations of damping. One such scheme for adding damping is the Rayleigh damping method which postulates that the damping matrix must be a linear combination of the mass and

the stiffness matrices:

$$\boldsymbol{C} = a_0\boldsymbol{M} + a_1\boldsymbol{L} \tag{A.33}$$

The coefficients a_0 and a_1 can be chosen such that certain damping ratios η_1 and η_2 are observed for frequencies f_1 and f_2. It can be shown that if the frequencies and damping ratios are known, then the coefficients are given by:

$$a_0 = \frac{4\pi f_1 f_2}{f_1^2 - f_2^2}\left(f_1\eta_2 - f_2\eta_1\right) \tag{A.34}$$

$$a_1 = \frac{f_1\eta_1 - f_2\eta_2}{\pi\left(f_1^2 - f_2^2\right)} \tag{A.35}$$

While the damping for other frequencies f can be evaluated to be:

$$\eta(f) = \frac{a_0}{2\pi f} + \pi a_1 f \tag{A.36}$$

Rayleigh's damping matrix has the advantage of being physically realizable as it yields a matrix in which connected degrees-of-freedom are assigned damping. On the other hand, this form of damping over-damps low frequencies (as can be seen from Equation A.36 by a diverging damping ratio for f values approaching zero) and also for very high frequencies.

Other damping methods exist with better frequency damping behavior, like Caughey damping matrices which extend Rayleigh's method for more than two target frequencies, or modal damping matrix which assigns per-mode frequencies. Both these methods have the drawback of yielding full matrices which are non-physical because non-connected degrees-of-freedom become causally coupled where no physical mechanism for such coupling exists. In reality the damping is caused by other forms of damping such as material yielding, damage, cracking, creeping and such, and are better explained by explicitly incorporating these effects into the material constitutive models.

Once the discretized equation of motion, damped or not, is established it must be solved by using some discrete dynamical integrator. Commonly, Newmark's method (Newmark, 1959) is used to solve for the increment in displacements, velocities and accelerations. A particular

application of Newmark's method within one step leads to the linear system of equations:

$$\left(\frac{1}{\beta\Delta t^2}\boldsymbol{M} + \frac{\gamma}{\Delta t}\boldsymbol{C} + \boldsymbol{K}^t\right)\Delta\boldsymbol{u} = \boldsymbol{R} \tag{A.37}$$

Where Δt is the time-step chosen, γ and β are parameters of the Newmark integrator, the residual $\boldsymbol{R} = \Delta\boldsymbol{F} - \Delta\boldsymbol{F}^{int} - \boldsymbol{C}\Delta\dot{\boldsymbol{u}} - \boldsymbol{M}\Delta\ddot{\boldsymbol{u}}$ and $\Delta\boldsymbol{F}^{int}$ represents the projection of the internal resisting force increments on the elements back into the global force vector. Once the increments in displacements $\Delta\boldsymbol{u}$ are obtained, the increments in velocities and accelerations come from.

$$\Delta\dot{\boldsymbol{u}} = \frac{\gamma}{\beta\Delta t}\Delta\boldsymbol{u} \qquad\qquad \Delta\ddot{\boldsymbol{u}} = \frac{1}{\beta\Delta t^2}\Delta\boldsymbol{u} \tag{A.38}$$

Initially the increments in internal resisting forces, $\Delta\boldsymbol{F}^{int}$, are unknown. It is assumed that they are zero and Equation A.37 is solved, $\Delta\boldsymbol{F}^{int}$ are evaluated and it is checked whether Equation A.37 is satisfied within a given tolerance. If it is not satisfied, then the system is solved again (this time using the previous value of $\Delta\boldsymbol{F}^{int}$) and new residual is computed and checked for convergence. This procedure is known as Newton iteration, and it's convergence is not guaranteed. There exists other schemes for solving the non-linear system of equations which are out of the scope of this review. Notably, though, it is possible to avoid iteration by using smaller time-steps although this results in error accumulation with time. If this strategy is chosen, it is important to show convergence by using two time steps and showing that results don't change significantly between.

For whatever solution method is chosen, once convergence has been achieved the displacement increment $\delta\boldsymbol{u}$ is accepted and used to compute increments strain and stress at all material Gauss-points though Equation A.15 and the material constitutive rate equations, Equation A.20. The new material state will be used to compute material tangent moduli at the beginning of the next time step. And so the simulation is continued until all time-steps have been completed.

APPENDIX B

Example of Analysis Using the FEI

B.1. Triaxial test

The first example is a sigle element simulation meant to test a constitutive model. In this case we are testing the 'Pisano' (Pisanò and Jeremić, 2014) model for cyclic behavior of soils, within an eight node brick with three degrees-of-freedom per node and eight Gauss point quadrature integration rule. The test setup is shown in B.1, where a single eight-node-brick is constrained and confined as shown.

The modeling scripts starts with a model name and the setup of some general model parameters which are used in the rest of the script. These are shown in B.2.

The material model to be used in this test is added next in B.3. Please, note the use of some of the parameters defined in the previous section as input to this model. The Pisano model is a newly developed constitutive model for soil modeling based on boundary surface plasticity and vanishing elastic region concept. It was developed specificly to match $G/G_{\max}$ and

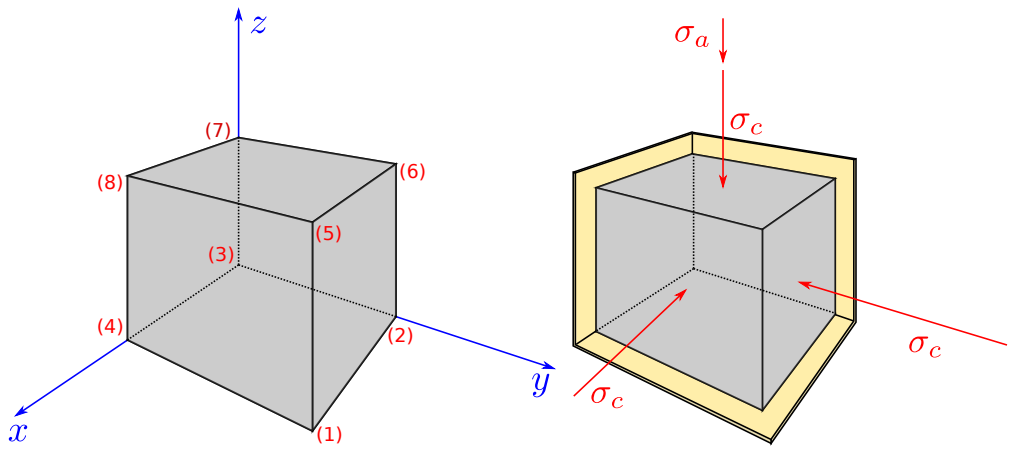


FIGURE B.1. Single element triaxial test model setup.

```
1  model name "Triaxial_Test";
2
3  nu         = 0.25;         // Poisson's ratio
4  G_max      = 4*MPa;        // Maximum shear modulus
5  p0         = 80*kPa;       // Initial confinement
6  gamma_max  = 0.1;          // Maximum shear deformation
7  Bulk       = 2*G_max*(1+nu)/(3*(1-2*nu)); // Bulk modulus
8  sigma_c    = -40*kPa;      // Triaxial chamber pressure
9  max_disp   = 0.1*cm;       // Maximum displacement
10 disp_rate  = 0.0001*mm/s; // Rate of displacement
11 dx         = 1.0*m;        // Length of sample in x
12 dz         = 1.0*m;        // Length of sample in z
13 dy         = 1.0*m;        // Length of sample in y
```

FIGURE B.2. Example 1, Triaxial test. Definition of model parameters.

damping curves for soil, while retaining enough flexibility to provide dilatancy and Lode angle dependency of the strength.

```
15 add material # 1 type pisano
16   elastic_modulus = G_max*2*(1+nu)
17   poisson_ratio = nu
18   M_in = 1.2
19   kd_in = 0.0
20   xi_in = 0.0
21   h_in = G_max/(1.5*p0)
22   m_in = 1.0
23   mass_density = 0*kg/m^3
24   initial_confining_stress = p0
25   beta_min = 0.0;
```

FIGURE B.3. Example 1, Triaxial test. Adding the material to the domain.

Next, in B.4 the 8 nodes that make up a brick element are added along with the single element in this example. Note that the coordinates of the nodes are given by using the variable set in the beginning of the script, along with regular quantities defined in place.

The boundary conditions are defined in B.5, note the use of the reserved keywords to refer to the degrees-of-freedom being constrained.

The first stage of loading is started at line 48 on B.6, in which the model is subjected to the chamber pressure. This is done by applying this pressure as forces on the nodes of the free faces.

```
27 add node # 1 at (0*m, 0*m, 0*m) with 3 dofs;
28 add node # 2 at ( dx, 0*m, 0*m) with 3 dofs;
29 add node # 3 at ( dx,  dy, 0*m) with 3 dofs;
30 add node # 4 at (0*m,  dy, 0*m) with 3 dofs;
31 add node # 5 at (0*m, 0*m,  dz) with 3 dofs;
32 add node # 6 at ( dx, 0*m,  dz) with 3 dofs;
33 add node # 7 at ( dx,  dy,  dz) with 3 dofs;
34 add node # 8 at (0*m,  dy,  dz) with 3 dofs;
35
36 add element # 1 type 8NodeBrick
37   with nodes ( 5, 6, 7, 8, 1, 2, 3, 4) use material # 1;
```

FIGURE B.4. Example 1, Triaxial test. Definition of eight nodes and the brick element.

```
40 fix node # 1 dofs ux uy uz;
41 fix node # 2 dofs    uy uz;
42 fix node # 3 dofs       uz;
43 fix node # 4 dofs ux    uz;
44 fix node # 5 dofs ux uy   ;
45 fix node # 6 dofs    uy   ;
46 fix node # 8 dofs ux      ;
```

FIGURE B.5. Example 1, Triaxial test. Boundary conditions

Next, lines 74 to 82 on B.8 show the analysis options chosen for this model. Load factor increment is defined on line 79 to be 0.01, which means that the target chamber pressure will be obtained after 100 static steps. Therefore the analysis is run, on line 82, with 100 steps.

After this consolidations stage the model is ready for shearing. This is accomplished by adding additional loads on the top face of the brick. B.8 starts on line 85 with a new stage declaration. This is important because when a new stage is started, all loads of the previous stages are set to be constant, that is, to not increase with analysis time step. Also the loading in this case will be applied through displacements. Note the use of a `while` loop to apply an imposed motion (contained in separate input text files) to the nodes involved in the top face. The loading needs to be applied in this fashion because of plastification which might lead to an unstable system of equations.

Finally, the program exits with the `bye;` statement.

```
new loading stage "consolidation";

A1 = dy*dz;
A2 = dx*dz;
A3 = dx*dy;

Force1 = -sigma_c/4*A1;
Force2 = -sigma_c/4*A2;
Force3 = -sigma_c/4*A3;

add load #  1 to node # 5 type linear Fz = Force3;
add load #  2 to node # 6 type linear Fz = Force3;
add load #  3 to node # 7 type linear Fz = Force3;
add load #  4 to node # 8 type linear Fz = Force3;

add load #  5 to node # 2 type linear Fx = Force1;
add load #  6 to node # 3 type linear Fx = Force1;
add load #  7 to node # 6 type linear Fx = Force1;
add load #  8 to node # 7 type linear Fx = Force1;

add load #  9 to node # 3 type linear Fy = Force2;
add load # 10 to node # 4 type linear Fy = Force2;
add load # 11 to node # 7 type linear Fy = Force2;
add load # 12 to node # 8 type linear Fy = Force2;
```

FIGURE B.6. Example 1, Triaxial test. Confinement stage.

```
define convergence test Norm_Displacement_Increment
  tolerance          = 1e-8*mm
  maximum_iterations = 100
  verbose_level      = 2;
define algorithm Newton;
define load factor increment 0.01;
define solver UMFPack;

simulate 100 steps using static algorithm;
```

FIGURE B.7. Example 1, Triaxial test. Simulation options.

B.2. 1–D Wave propagation

Example listings B.9, B.10, and B.11 demonstrate the usage of the DSL for 1–D SH-wave propagation though a column of linear-elastic-isotropic soil with varying shear wave velocity with depth. The example shows how the language can be used to create a mesh of elements and materials programatically.

```
85  new loading stage "shearing";
86
87  n = 5;
88  while (n <= 8) // To apply sigma_a
89  {
90    add imposed motion # i to node # n dof uz
91      time_step                = 1*s
92      displacement_scale_unit  = -max_disp
93      displacement_file        = "input_displacement.dat"
94      velocity_scale_unit      = test_velocity
95      velocity_file            = "input_velocity.dat"
96      acceleration_scale_unit  = 0*g
97      acceleration_file        = "input_acceleration.dat";
98    n += 1;
99    i += 1;
100 };
101
102 define convergence test Norm_Displacement_Increment
103   tolerance          = 1e-8*mm
104   maximum_iterations = 100
105   verbose_level      = 2;
106 define algorithm Newton;
107 define load factor increment 0.01;
108 define solver UMFPack;
109
110 simulate 100 steps using static algorithm;
111
112 bye;
```

FIGURE B.8. Example 1, Triaxial test. Second stage.

Lines 1 to 14 declare parameters that control the simulation. Note that this part of the code controls the number of elements that will be generated, as well as the distribution of stiffness in depth.

Lines 18-26 add the bottom four nodes constrain them to move only in the X direction. On Lines 28-60 there is a `while` loop that programatically generates the column of elements. The first four lines inside the loop determine the material properties (from linear interpolation) for the element to be generated. Lines 38-46 generate new nodes and constrain them to move only in the X direction. Lines 48-51 contain the material definition, using the properties generated previously. Finally, lines 53-56 create an element connecting the new nodes with the ones generated on the previous step (or outside the loop for the first iteration).

```
1  model name "wave_propagation_example";
2
3  Height      = 1000*m;
4  NBricks     = 100;
5  max_disp    = .0001*mm;
6  dt          = 0.01*s;
7  Nsteps      = 900;
8  rho_bottom  = 2200*kg/m^3;
9  rho_top     = 2000*kg/m^3;
10 Vs_bottom   = 1500*m/s;
11 Vs_top      = 500*m/s;
12 nu          = 0.25;
13
14 dz          = Height / NBricks;
```

FIGURE B.9. Example 2, 1–D Wave propagation simulation. This listing set the global parameters that control the simulation.

Next, on B.11, lines 63-77 are used define a new loading stage called 'earthquake' and to set the imposed input motion (defined in separate text files) for the bottom nodes. Lines 81-91 define the simulation parameters: using a Newmark dynamic integration method, etc. Finally, the simulation is executed on lines 93 and 94 with the `simulate` command.

```
18 add node # 1 at (1*m, 1*m , 0*m) with 3 dofs;
19 add node # 2 at (0*m, 1*m , 0*m) with 3 dofs;
20 add node # 3 at (0*m, 0*m , 0*m) with 3 dofs;
21 add node # 4 at (1*m, 0*m , 0*m) with 3 dofs;
22
23 fix node No 1 dofs uy uz;
24 fix node No 2 dofs uy uz;
25 fix node No 3 dofs uy uz;
26 fix node No 4 dofs uy uz;
27
28 mid_height_current_layer = dz/2;
29 e = 0;
30 while ( e < NBricks)
31 {
32     rho = rho_bottom + (rho_top - rho_bottom)/Height * ↩
          mid_height_current_layer;
33     Vs  = Vs_bottom + (Vs_top - Vs_bottom)/Height * ↩
          mid_height_current_layer;
34
35     G = (rho*Vs^2)*(0.5);
36     E = 2*G*(1+nu);
37
38     add node # (5+4*e) at (1*m, 1*m , dz*(e+1) ) with 3 dofs;
39     add node # (6+4*e) at (0*m, 1*m , dz*(e+1) ) with 3 dofs;
40     add node # (7+4*e) at (0*m, 0*m , dz*(e+1) ) with 3 dofs;
41     add node # (8+4*e) at (1*m, 0*m , dz*(e+1) ) with 3 dofs;
42
43     fix node # (5+4*e) dofs uy uz;
44     fix node # (6+4*e) dofs uy uz;
45     fix node # (7+4*e) dofs uy uz;
46     fix node # (8+4*e) dofs uy uz;
47
48     add material #  type linear_elastic_isotropic_3d
49         mass_density = rho
50         elastic_modulus = E
51         poisson_ratio = nu;
52
53     add element # e+1 type 8NodeBrickLT with nodes
54         (4*e+5, 4*e+6, 4*e+7, 4*e+8,
55          4*e+1, 4*e+2, 4*e+3, 4*e+4)
56         use material # e+1;
57
58     e += 1;
59     mid_height_current_layer += dz;
60 }
```

FIGURE B.10. Example 2, 1–D Wave propagation simulation. Automatic generation of material properties, nodes, elements, and constrains based on global parameters.

```
new loading stage "earthquake";

n = 1;
while(n <= 4)
{
    add imposed motion # n to node # n dof ux
            time_step = dt
            displacement_scale_unit = max_disp
            displacement_file = "disp.dat"
            velocity_scale_unit = m/s
            velocity_file = "disp.dat"
            acceleration_scale_unit = m/s/s
            acceleration_file = "disp.dat";
    n += 1;
}

define dynamic integrator Newmark with
    gamma = 0.5
    beta = 0.25;
define convergence test Norm_Displacement_Increment
    tolerance = 0.001*m
    maximum_iterations = 100
    verbose_level = 0;
define algorithm Newton;
define solver ProfileSPD;

simulate 3000 steps using transient algorithm
    time_step = dt;

bye;
```

FIGURE B.11. Example 2, 1–D Wave propagation simulation. Input motion is set as a 'imposed motion' boundary condition and the simulation is setup and executed

TABLE B.1. List of predefined units in FEM

Unit	Name	Unit of	Scale	SI Unit
`m`	meter	Length	1	m
`s`	second	Time	1	s
`kg`	kilogram	Mass	1	kg
`Km`	kilometer	Length	1×10^{3}	m
`cm`	centimeter	Length	1×10^{-2}	m
`mm`	millimeter	Length	1×10^{-3}	m
`ms`	millisecond	Time	1×10^{-3}	s
`s`	micro-second	Time	1×10^{-6}	s
`ns`	nano-second	Time	1×10^{-9}	s
`Hz`	Hertz	Frequency	1×10^{-1}	s^{-1}
`g`	Gravitational constant	Acceleration	9.81×10^{1}	$m \cdot s^{-2}$
`N`	Newton	Force	1×10^{1}	$kg \cdot m \cdot s^{-2}$
`kN`	Kilo-Newton	Force	1×10^{3}	$kg \cdot m \cdot s^{-2}$
`Pa`	Pascal	Stress	1×10^{1}	$kg \cdot m^{-1} \cdot s^{-2}$
`kPa`	Kilo-Pascal	Stress	1×10^{3}	$kg \cdot m^{-1} \cdot s^{-2}$
`GPa`	Giga-Pascal	Stress	1×10^{6}	$kg \cdot m^{-1} \cdot s^{-2}$
`MPa`	Mega-Pascal	Stress	1×10^{9}	$kg \cdot m^{-1} \cdot s^{-2}$
`pi`			3.14159	

TABLE B.2. Second-order keywords defined in FEDEASLab

Keyword type	Keywords available
element_type	8NodeBrick, 8NodeBrickLT, 20NodeBrick, 27NodeBrick, 27NodeBrickLT, 8NodeBrick_elastic, 20NodeBrick_elastic, 27NodeBrick_elastic, 8NodeBrick_upU, 8NodeBrick_up, 20NodeBrick_upU, beam_displacement_based, beam_elastic, beam_9dof_elastic, beam_elastic_lumped_mass, 4NodeShell_MITC4, 4NodeShell_NewMITC4, 3NodeShell_ANDES, 4NodeShell_ANDES, truss, penalty, penalty_for_applying_generalized_displacement, contact, variable_node_brick_8_to_27
material_type	linear_elastic_isotropic_3d, VonMisesLT, DruckerPragerLT, DruckerPragerVonMisesLT, DruckerPragerArmstrongFrederickLT, sanisand2008, camclay, camclay_accelerated, sanisand2004, linear_elastic_crossanisotropic, uniaxial_elastic, uniaxial_steel01, uniaxial_steel02, uniaxial_concrete02, linear_elastic_isotropic_3d_LT, PisanoLT
dynamic_integrator_keyword	Newmark, HHT
static_integrator_keyword	LoadControl
convergence_test_keyword	Norm_Displacement_Increment, Norm_Energy_Increment, Norm_Unbalance, Arc_Length
algorithm_keyword	Newton, Modified_Newton, With_No_Convergence_Check
solver_keyword	ProfileSPD, UMFPack, Parallel
force_keyword	Fx, Fy, Fz, Mx, My, Mz, F_fluid_x, F_fluid_y, F_fluid_z

Bibliography

Abell, J. visitESSI plugin [2015].

Abrahams, D. and *Gurtovoy, A. C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond (C++ in Depth Series)*. Addison-Wesley Professional [2004]. ISBN 0321227255.

Aki, K. and *Richards, P. G. Quantitative Seismology*. University Science Books, 2^{\rm nd} edition [2002].

Alvin, K., *de la Fuente, H. M.*, *Haugen, B.*, and *Felippa, C. A.* Membrane Triangles with corner drilling freedoms – {I}. the {EFF} element. *Finite Elem. Anal. Des.*, **12**:pages 163–187 [1992].

Armstrong, P. J., *Frederick, C. O.*, and *Armstrong, P. J.* A mathematical representation of the multiaxial Bauschinger effect. Technical Report RD/B/N/ 731,, C.E.G.B. [1966]. doi: 10.3184/096034007x207589.

Balay, S., *Abhyankar, S.*, *Adams, M.*, *Brown, J.*, *Brune, P.*, *Buschelman, K.*, *Dalcin, L.*, *Eijkhout, V.*, *Gropp, W.*, *Kaushik, D.*, *Knepley, M.*, *McInnes, L. C.*, *Rupp, K.*, *Smith, B.*, *Zampini, S.*, and *Zhang, H.* {PETS}c Users Manual. Technical Report ANL-95/11 - Revision 3.6, Argonne National Laboratory [2015a].

Balay, S., *Abhyankar, S.*, *Adams, M.*, *Brown, J.*, *Brune, P.*, *Buschelman, K.*, *Dalcin, L.*, *Eijkhout, V.*, *Gropp, W.*, *Kaushik, D.*, *Knepley, M.*, *McInnes, L. C.*, *Rupp, K.*, *Smith, B.*, *Zampini, S.*, and *Zhang, H.* {PETS}c {W}eb page. \url{http://www.mcs.anl.gov/petsc} [2015b].

Balay, S., *Gropp, W. D.*, *McInnes, L. C.*, and *Smith, B. F.* Efficient Management of Parallelism in Object Oriented Numerical Software Libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Mod. Softw. Tools Sci. Comput.*, pages 163–202. Birkh{ä}user Press [1997].

Bergan, P. G. and *Felippa, C. A.* A Triangular Membrane Element with Rotational Degrees of Freedom. *Comput. Methods Appl. Mech. Eng.*, **50**:pages 25–69 [1985].

Bielak, J., *Loukakis, K.*, *Hisada, Y.*, and *Yoshimura, C.* Domain Reduction Method for Three–Dimensional Earthquake Modeling in Localized Regions. Part {I}: Theory. *Bull. Seismol. Soc.*

Am., **93(2)**:pages 817–824 [2003].

Borja, R. I., *Chao, H.-Y.*, *Montáns, F. J.*, and *Lin, C.-H.* Nonlinear ground response at Lotung LSST site. *J. Geotech. geoenvironmental Eng.*, **125(3)**:pages 187–197 [1999].

Boulanger, R., *Curras, C.*, *L.Kutter, B.*, *Wilson, D. W.*, and *Abghari, A.* Seismic Soil-Pile-Structure Interaction Experiments and Analyses. *J. Geotech. geoenvironmental Eng.*, **125(September)**:pages 750–759 [1999]. doi:10.1061/(ASCE)1090-0241(1999)125.

Brune, J. N. Tectonic stress and the spectra of seismic shear waves from earthquakes. *J. Geophys. Res.*, **75(26)**:page 4997 [1970]. ISSN 0148-0227. doi:10.1029/JB075i026p04997.

Brune, J. N. Correction. *J. Geophys. Res.*, **76(20)**:page 5002 [1971]. ISSN 2156-2202. doi:10.1029/JB076i020p05002.

Burridge, R. and *Knopoff, L.* Body force equivalents for seismic dislocations. *Bull. Seismol. Soc. Am.*, **54(6A)**:pages 1875–1888 [1964].

Cecilio, Lira, D., *Bernard Devloo, P. R.*, and *Raggio Santos, E. S.* On an object oriented implementation of plasticity models. In E. Dvorkin, M. Goldschmit, and M. Storti, editors, *Mecánica Comput.*, volume XXIX, pages 4235–4245. Buenos Aires, Argentina [2010].

Childs, H., *Brugger, E.*, *Whitlock, B.*, *Meredith, J.*, *Ahern, S.*, *Pugmire, D.*, *Biagas, K.*, *Miller, M.*, *Harrison, C.*, *Weber, G. H.*, *Krishnan, H.*, *Fogal, T.*, *Sanderson, A.*, *Garth, C.*, *Bethel, E. W.*, *Camp, D.*, *Rübel, O.*, *Durant, M.*, *Favre, J. M.*, and *Navrátil, P.* VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In *High Perform. Vis. Extrem. Sci. Insight*, pages 357–372 [2012].

Courant, Richard; Friedrichs, Kurt; Lewy, H. On the partial difference equations of mathematical physics. Technical report, Courant Institute of Mathematical Sciences, New York University, New York, NY, U.S.A. [1956].

Crempien, J. G. F. and *Archuleta, R. J.* UCSB Method for Simulation of Broadband Ground Motion from Kinematic Earthquake Sources. *Seismol. Res. Lett.*, **86(1)**:pages 61–67 [2015]. ISSN 0895-0695. doi:10.1785/0220140103.

Crisfield, M. Non-linear finite element analysis of solids and structures, Volume 1. John Wiley & Sons, West Sussex, first edition [1991]. ISBN 0 471 92956 5.

Dreger, D. S., *Huang, M.-H.*, *Rodgers, A.*, *Taira, T.*, and *Wooddell, K.* Kinematic Finite-Source Model for the 24 August 2014 South Napa, California, Earthquake from Joint Inversion of Seismic,

GPS, and InSAR Data. *Seismol. Res. Lett.*, **(August 2014)** [2015]. ISSN 0895-0695. doi:10.1785/0220140244.

Einstein, A. The Foundation of the General Theory of Relativity. *Ann. Phys.*, **49**:pages 769–822 [1916]. doi:10.1002/andp.200590044.

Felippa, C. A. and *Alexander, S.* Membrane Triangles with corner drilling freedoms – {III}. Implementation and Performance Evaluation. *Finite Elem. Anal. Des.*, **12**:pages 203–239 [1992].

Felippa, C. A. and *Militello, C.* Membrane Triangles with corner drilling freedoms – {II}. the {ANDES} element. *Finite Elem. Anal. Des.*, **12**:pages 189–201 [1992].

Fortuño, C., *de la Llera, J.*, *Wicks, C.*, and *Abell, J.* Synthetic Hybrid Broadband Seismograms Based on InSAR Coseismic Displacements. *Bull. Seismol. Soc. Am.*, **104(6)**:pages 2735–2754 [2014]. ISSN 0037-1106. doi:10.1785/0120130293.

Geuzaine, C. and *Remacle, J.-F.* {Gmsh}: A {3-D} finite element mesh generator with built-in pre- and post-processing facilities. *Int. J. Numer. Methods Eng.*, **79(11)**:pages 1309–1331 [2009]. ISSN 00295981. doi:10.1002/nme.2579.

He, Y.-M., *Wang, W.-M.*, and *Yao, Z.-X.* Static Deformation Due to Shear and Tensile Faults in a Layered Half-Space. *Bull. Seism. Soc. Am.*, **93(5)**:pages 2253–2263 [2003]. doi:10.1785/0120020136.

Hughes, T. J. R. The Finite Element Method. Dover Publications Inc., Englewood Cliffs, NJ [2000]. ISBN 0-486-41181-1.

Isbiliroglu, Y., *Taborda, R.*, and *Bielak, J.* Coupled Soil-Structure Interaction Effects of Building Clusters During Earthquakes. *Earthq. Spectra*, **31(1)**:pages 463–500 [2015]. ISSN 87552930. doi:http://dx.doi.org/10.1193/102412EQS315M.

Jeremić, B. and *Jie, G.* Plastic Domain Decomposition Method for Parallel Elastic–Plastic Finite Element Computations in Gomechanics. Technical Report UCD–CompGeoMech–03–07, University of California Davis, Davis [2007].

Jeremić, B., *Jie, G.*, *Preisig, M.*, and *Tafazzoli, N.* Time domain simulation of soil-foundation-structure interaction in non-uniform soils. *Earthq. Eng. Struct. Dyn.*, **38(5)**:pages 699–718 [2009]. ISSN 00988847. doi:10.1002/eqe.896.

Jeremić, B., *Tafazzoli, N.*, *Ancheta, T.*, *Orbović, N.*, and *Blahoianu, A.* Seismic behavior of NPP structures subjected to realistic 3D, inclined seismic motions, in variable layered soil/rock, on surface or embedded foundations. *Nucl. Eng. Des.*, **265**:pages 85–94 [2013]. ISSN 00295493. doi:10.1016/j.nucengdes.2013.07.003.

Jeremić, B. and *Yang, Z.* Template Elastic–Plastic Computations in Geomechanics. *Int. J. Numer. Anal. Methods Geomech.*, **26(14)**:pages 1407–1427 [2002].

Ji, C., *Wald, D. J.*, and *Helmberger, D. V.* Source Description of the 1999 Hector Mine , California , Earthquake , Part II : Complexity of Slip History. *Society*, **92(4)**:pages 1208–1226 [2002a].

Ji, C., *Wald, D. J.*, and *Helmberger, D. V.* Source Description of the 1999 Hector Mine, California, Earthquake, Part I: Wavelet Domain Inversion Theory and Resolution Analysis. *Seism. Soc. Am*, **92(4)**:pages 1192–1207 [2002b]. ISSN 0037-1106. doi:10.1785/0120000916.

Jirasek, M. and *Bazant, Z. P.* General Elastoplastic Constitutive Models. *Inelast. Anal. Struct.* [2001].

Kanamori, H. The energy release in great earthquakes. *J. Geophys. Res.*, **82(20)**:page 2981 [1977]. ISSN 0148-0227. doi:10.1029/JB082i020p02981.

Karypis, G. and *Kumar, V.* Multilevel Graph Partitioning Schemes. *Int. Conf. Parallel Process.*, **3**:pages 113–122 [1995]. ISSN 01903918. doi:10.1145/224170.224229.

Kontoe, S., *Zdravkovic, L.*, *Menkiti, C. O.*, and *Potts, D. M.* Seismic response and interaction of complex soil retaining systems. *Comput. Geotech.*, **39**:pages 17–26 [2012]. ISSN 0266352X. doi: 10.1016/j.compgeo.2011.08.003.

Kramer, S. L. Geotechnical Earthquake Engineering. Prentice Hall, Seattle, WA, 1 edition [1996]. ISBN 0-13-374943-6.

Limache, A. and *Rojas Fredini, P.* A tensor library for scientific computing. In A. Cardona, M. Storti, and C. Zuppa, editors, *Mecánica Comput.*, volume XXVII, pages 2907–2925. San Luis, Argentina [2008].

Lubarda, V. A. and *Benson, D. J.* On the numerical algorithm for isotropic-kinematic hardening with the Armstrong-Frederick evolution of the back stress. *Comput. Methods Appl. Mech. Eng.*, **191(33)**:pages 3583–3596 [2002]. ISSN 00457825. doi:10.1016/S0045-7825(02)00296-7.

Lymser, J. {SASSI}: A computer program for dynamic soil structure interaction analysis. *Rep. UBC/GT81-02. Univ. California, Berkeley, CA, USA.* [1988].

Mai, P. M. and *Beroza, G. C.* Source Scaling Properties from Finite-Fault-Rupture Models. *Bull. Seism. Soc. Am.*, **90(3)**:pages 604–615 [2000]. doi:10.1785/0119990126.

Mejia, L. H. and *Dawson, E. M.* Earthquake deconvolution for FLAC. *Proc. 4th Int. FLAC Symp. Numer. Model. Geomech.*, **(1969)**:pages 1–9 [2006].

Menbtrey, P., *Zimmermann, T.*, *Menéntrey, P.*, and *Zimmermann, T.* Object-Oriented Non-Linear Finite Element Analysis : Application To J2 Plasticity. *Comput. Struct.*, **49(5)**:pages 767–777 [1993].

Mernik, M., *Heering, J.*, and *Sloane, A. M.* When and how to develop domain-specific languages. *ACM Computing Surveys*, **37(4)**:pages 316–344 [2005]. `http://doi.acm.org/10.1145/1118890.1118892`.

NEHRP Consultants Joint Venture. Soil-Structure Interaction for Building Structures. Technical report, National Institute of Standards and Technology, Gaithersburg, MD [2012].

Newmark, N. M. A Method for Structural Dynamics. *J. Eng. Mech. Div. ASCE*, **85(3)**:pages 67–94 [1959].

Newton, I., *Motte, A.*, and *Machin, J. The Mathematical Principles of Natural Philosophy*. Number v. 1 in The Mathematical Principles of Natural Philosophy. B. Motte [1729].

Okada, Y. Surface deformation due to shear and tensile faults in a half-space Okada, Y Bull Seismol Soc AmV75, N4, Aug 1985, P1135–1154. *Int. J. Rock Mech. Min. Sci. Geomech. Abstr.*, **23(4)**:pages 128–128 [1986]. ISSN 01489062. doi:10.1016/0148-9062(86)90674-1.

Oppenheim, A. V., *Willsky, A. S.*, and *Nawab, S. H. Signals and Systems*. Prentice-Hall signal processing series. Prentice Hall [1997]. ISBN 9780138147570.

Petersson, N. A. Stable and Efficient Modeling of Anelastic Attenuation in Seismic Wave Propagation. *Commun. Comput. Phys.*, **12(1)**:pages 193–225 [2012]. ISSN 18152406. doi:10.4208/cicp.201010.090611a.

Petersson, N. A. and *Sjogreen, B.* Super-grid Modeling of the Elastic Wave Equation in Semi-bounded Domains. **955**:pages 913–955 [2013].

Peyrat, S., *Olsen, K.*, and *Madariaga, R.* Dynamic modeling of the 1992 Landers earthquake. *J. Geophys. Res.*, **106(B11)**:page 26467 [2001]. ISSN 0148-0227. doi:10.1029/2001JB000205.

Pisanò, F. and *Jeremić, B.* Simulating stiffness degradation and damping in soils via a simple visco-elastic–plastic model. *Soil Dyn. Earthq. Eng.*, **63**:pages 98–109 [2014]. ISSN 02677261. doi:10.1016/j.soildyn.2014.02.014.

Psarropoulos, P. N., *Tazoh, T.*, *Gazetas, G.*, and *Apostolou, M.* Linear and Nonlinear Valley Amplification Effects of Seismic Ground Motion. *Soils Found. Japanese Geotech. Soc.*, **47(5)**:pages 857–871 [2007].

Rodgers, A. and *Pitarka, A.* 3D Ground Motion Simulations of the 2014 South Napa Earthquake using the USGS Geologic / Seismic Model and Various Source Models. In *Seism. Soc. Am*, APRIL [2015].

Schmedes, J., *Archuleta, R. J.*, and *Lavallée, D.* Correlation of earthquake source parameters inferred from dynamic rupture simulations. *J. Geophys. Res. Solid Earth*, **115(3)**:pages 1–12 [2010]. ISSN 21699356. doi:10.1029/2009JB006689.

Schmedes, J., *Archuleta, R. J.*, and *Lavallee, D.* A kinematic rupture model generator incorporating spatial interdependency of earthquake source parameters. *Geophys. J. Int.*, **192(3)**:pages 1116–1131 [2013]. ISSN 0956540X. doi:10.1093/gji/ggs021.

Scholz, C. H. The Mechanics of Earthquakes and Faulting. Cambridge University Press [2002]. ISBN 9780521655408.

Semblat, J. Rheological Interpretation of Rayleigh Damping. *J. Sound Vib.*, **206(5)**:pages 741–744 [1997]. ISSN 0022460X. doi:10.1006/jsvi.1997.1067.

Sjögreen, B. and *Petersson, N. A.* A Fourth Order Accurate Finite Difference Scheme for the Elastic Wave Equation in Second Order Formulation. *J. Sci. Comput.*, **52(1)**:pages 17–48 [2011]. ISSN 0885-7474. doi:10.1007/s10915-011-9531-1.

Spinellis, D. Notable design patterns for domain-specific languages. *Journal of Systems and Software*, **56(1)**:pages 91–99 [2001]. `http://dx.doi.org/10.1016/S0164-1212(00)00089-3`.

Stewart, J. P., *Fenves, G. L.*, and *Seed, R. B.* Seismic soil-structure interaction in buildings. I: Analytical aspects. *J. Geotech. Geoenv. Engrg*, **125(1)** [1999a].

Stewart, J. P., *Seed, R. B.*, and *Fenves, G. L.* Seismic soil-structure interaction in buildings. II: Empirical findings. *J. Geotech. Geoenv. Engrg.*, **125(1)** [1999b].

Stock, C. and *Smith, E.* Evidence for different scaling of earthquake source parameters for large earthquakes depending on faulting mechanism. *Geophys. J. Int.*, **143**:pages 157–162 [2000]. doi: 10.1046/j.1365-246x.2000.00225.x.

Tafazzoli, N. Methods, Computational Platform, Verification, and Application of Earthquake-Soil-Structure-Interaction Modeling and Simulation. Phd, University of California at Davis [2012].

Tripe, R., *Kontoe, S.*, and *Wong, T. K. C.* Slope topography effects on ground motion in the presence of deep soil layers. *Soil Dyn. Earthq. Eng.*, **50**:pages 72–84 [2013]. ISSN 02677261. doi:10.1016/j.soildyn.2013.02.011.

Yoshimura, C., *Bielak, J.*, and *Hisada, Y.* Domain Reduction Method for Three–Dimensional Earthquake Modeling in Localized Regions. Part {II}: Verif\/ication and Examples. *Bull. Seismol. Soc. Am.*, **93(2)**:pages 825–840 [2003].

Zabaras, N. and *Srikanth, A.* Using Objects to Model Finite Deformation Plasticity 2 . A Lagrangian Analysis for Large Deformations of Microporous Materials. pages 37–60 [1999].

Zhong, R. and *Huang, M.* Winkler model for dynamic response of composite caisson–piles foundations: Seismic response. *Soil Dyn. Earthq. Eng.*, **66**:pages 241–251 [2014]. ISSN 02677261. doi:10.1016/j.soildyn.2014.07.005.

Zhu, L. and *Rivera, L. a.* A note on the dynamic and static displacements from a point source in multilayered media. *Geophys. J. Int.*, **148(3)**:pages 619–627 [2002]. ISSN 0956540X. doi: 10.1046/j.1365-246X.2002.01610.x.