# DEVELOPMENT, VERIFICATION, AND VALIDATION OF A HYBRID TESTING FRAMEWORK FOR LATTICED STRUCTURES WITH NONLINEAR GEOMETRIC EFFECTS

by

Michael Allen Tedeschi

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Civil Engineering

Charlotte

2015

Approved by:

_____
Dr. Matthew J. Whelan

_____
Dr. David C. Weggel

_____
Dr. Brett Q. Tempest

# ABSTRACT

MICHAEL ALLEN TEDESCHI. Development, verification, and validation of a hybrid testing framework for latticed structures with nonlinear geometric effects. (Under the direction of DR. MATTHEW J. WHELAN)

Hybrid simulation is an increasingly popular experimental method for structural dynamics research, as it offers the realism and exploratory power of full-scale testing at a fraction of the cost and size by resolving a structural system into complementary experimental and analytical substructures to be analyzed simultaneously as part of a single simulation. In order to promote the expansion of hybrid simulation into new areas of applied structural dynamics research within the energy industry, a pseudo-dynamic hybrid simulation software framework was developed, numerically verified, and experimentally validated. This framework employs a combination of an iterative implicit integration scheme and an $\alpha$-operator splitting scheme to perform dynamic analysis of highly nonlinear latticed structures. Historically, iterative methods have been avoided in hybrid simulation, since physical iterations may produce unintended and irrecoverable plastic deformation of the test specimen within the iteration, thereby corrupting the fidelity of the hybrid simulation. The software framework implemented in this thesis enforces iterative displacements numerically rather than physically, thus avoiding experimental errors associated with the path dependency of the experimental substructure. In this work, the framework was numerically verified using a model of a simple planar truss subjected to harmonic loading and a model of a large space truss subjected to base excitation from an earthquake accelerogram. Analyses accounting for geometric and material nonlinearities both separately and in combination with

one another were verified by comparison of predicted displacement time histories to results generated through an application programming interface with a commercial finite element software. Considerations involved in effective verification of the software, including the effect of the time step size on the accuracy of the framework, are discussed. Following verification, the hybrid testing software framework was experimentally validated through simulations of a power transmission tower loaded dynamically by theoretical galloping of an ice-covered conductor and a ground wire. A hybrid simulation was first performed within the linear elastic range of the specimen material. Excitation amplitudes were then increased into a range producing nonlinear plastic behavior of the experimental substructure. The observed displacement time histories and response hysteresis exhibited strong correlation with results from numerical simulations, ultimately validating the implementation of the framework. Lastly, recommendations are made for future developments and expansions of the software.

# ACKNOWLEDGEMENTS

I would like to express my deepest appreciation for my committee chair Dr. Matthew J. Whelan for his technical insight, guidance, and generous personal investments of time and energy towards my research. He has consistently challenged me through words and example to never stop raising the standard to which I hold my own work.

I am deeply grateful to the Energy Production and Infrastructure Center (EPIC) for their financial support of my research through the EPIC Graduate Research Assistantship program. I would also like to thank the Charlotte Research Scholars program for giving me the opportunity to study experimental structural dynamics.

I cannot adequately express my gratitude to my parents for their selfless and loving financial and moral support throughout my entire education. I am humbled at the thought of how they have sacrificed for me.

I would also like to thank Dr. Youngjin Park for his technical guidance in the execution of physical tests, Dr. David C. Weggel for his technical and editorial critique of my thesis, Dr. Brett Q. Tempest, for his technical review and support, and my classmates for their guidance and feedback throughout the process of my research.

Finally, I would like to thank my family and friends for their continuous prayers, encouragement, wisdom, and companionship. I am grateful for their steadying influence through both the trials and the successes of the past two years. I thank God for the gift of these past two years, for this education, for the people around me, and for the strength and peace of mind to complete this work. I am not a self-made man.

TABLE OF CONTENTS

# LIST OF FIGURES

# CHAPTER 1: INTRODUCTION

Over the past thirty years, engineers in the fields of structural design and analysis have increasingly relied on the power of computers and finite element methods to predict both linear and nonlinear responses of structures subjected to dynamic loads. Concurrently, the efficiency and capabilities of computers and computational methods have grown exponentially. Despite this rapid development of technology, the phenomenological understanding of the behavior of complex structural systems and materials through their limit states remains a fundamental area of both basic research and applied testing; such research and testing are essential to facilitating further improvements in computational methods and verifying and validating new numerical models. One of the foremost aims of experimental structural dynamics is to develop a greater level of understanding of these phenomena through applied research and experimental observation. As the engineering community expands the knowledge base of structural dynamics through experimental testing, it becomes better-equipped to use the powerful analytical tools at its disposal to address pressing needs of society, with more accurate predictions of complex structural responses to dynamic loads.

Numerous societal challenges lie in the field of structural dynamics, including improving the resiliency of structures subject to earthquake and wind loads, mitigating man-made hazards due to blast and progressive collapse, and ensuring the resiliency

of the power grid, such as by reducing the effects of galloping ice-covered conductors on power transmission structures. One of the fundamental challenges associated with applied research and experimental observation of these large civil and energy infrastructure systems is the often prohibitive expense and logistics necessary to construct, instrument, and test through failure such large structures. Hybrid testing provides an innovative solution to this problem through the concept of substructuring: large-scale structures can be tested as a combination of numerical and physical "substructures," allowing researchers to physically model a portion of a structure and experimentally observe its behavior in the context of a larger system without having to physically model the entire structure. The result is a family of testing methods that are simultaneously efficient, cost-effective, realistic, and more easily implemented than full-scale testing of large structures. Although hybrid simulation presents a new set of challenges for experimental research, the continued development and advancement of this promising and increasingly popular technique is a significant step in the direction of progress in the field of experimental structural dynamics.

## 1.1 Anticipated Contributions of Research Efforts

The development, verification, and validation of a software framework for hybrid simulation of space trusses with distributed geometric nonlinearities is the focus of this thesis. The research presented here is intended to inform future efforts to expand the developed hybrid simulation software framework, as well as to increase the accessibility of this important experimental technique of hybrid simulation for more widespread implementation. In particular, the current work suggests the extension of

hybrid simulation to the study of critical energy infrastructure such as latticed power transmission towers. In the experimental demonstration of the developed framework, a general transmission tower model is adopted as the representative application space for the framework. Hybrid testing methodology is used to perform a simulation of the dynamic response of the tower to galloping conductor loads. The work presented in this thesis is anticipated to serve as a case study for development, verification, and validation of hybrid simulation methods across a wide range of applications.

## 1.2 Organization of Thesis

This thesis is outlined as follows:

- Chapter 2 provides an overview of pseudodynamic hybrid simulation and other emerging forms of hybrid simulation, as well as the prominent experimental simulation methods from which hybrid simulations have evolved. Additionally, a review of various applications where hybrid simulation has been employed is presented.

- Chapter 3 explains the theoretical background for nonlinear geometric analysis of truss systems and presents two numerical integration schemes implemented in the development of the hybrid simulation software framework, including a predictor-corrector method and an iterative implicit integration scheme.

- Chapter 4 describes the developed software framework and provides detailed explanation of the mechanics of the software routine. Considerations involved in various aspects of the implementation of the numerical integration scheme,

including the usage of numerical equilibrium iterations on the experimental substructure, are detailed.

- Chapter 5 presents numerical verification of the implementation of the developed software routine by comparison with an accepted commercial finite element analysis software package. Explanation of the methods used to perform effective verification of the developed software with an automated framework are presented.

- Chapter 6 demonstrates successful implementation of the hybrid simulation software framework in actual experiments involving both linear elastic and nonlinear material behavior. The results of the experiments are validated through comparison with nonlinear direct time-history integration finite element analyses.

- Chapter 7 provides conclusion and a review of the work presented in this thesis and recommendations for future work, suggested improvements to the software framework, and important considerations for implementing extended versions of the framework.

# CHAPTER 2: LITERATURE REVIEW

In recent decades, the field of experimental structural dynamics has experienced significant evolution in physical simulation methods and grown in its areas of study and application. Physical testing methods have grown substantially in complexity, propelled by the power of advanced instrumentation and controls systems as well as computational methods. This chapter presents a review of the most prominent methods of experimental simulation, a detailed summary of hybrid simulation, and an overview of the common areas of application for hybrid simulation.

## 2.1 Experimental Simulation Methods

Research on structures subjected to seismic loading is commonly conducted using experimental techniques, as earthquakes involve dynamic and often nonlinear analyses that are challenging to faithfully predict numerically. To date, most of the work in this area is centered around quasi-static testing (QST), shake table testing (STT), effective force testing (EFT), or pseudodynamic testing (PSD) of complete structural systems using models of various scales [Shao and Enyart, 2014]. Each of these testing methods is suitable only for complete structural systems and is based on the governing principle that external forces, $\mathbf{f}_{ext}$, and internal forces, $\mathbf{f}_{int}$, are always in equilibrium at each time step. However, each of these methods differ in the computation of applied external forces and handling of inertial and damping forces. Ultimately, the

methods are all rooted in the same fundamental concept but take varying approaches to simulating the discrete-time dynamic equation of motion

$$\mathbf{M}\ddot{\mathbf{x}}_n + \mathbf{C}\dot{\mathbf{x}}_n + \mathbf{K}\mathbf{x}_n = \mathbf{f}_{ext} \tag{2.1}$$

under different practical testing idealizations and assumptions.

Of the four testing methods mentioned above, QST is the most basic. In a QST, hydraulic actuators are typically used to quasi-statically apply a predefined cyclic displacement or force history [Shao and Enyart, 2014], as shown in Figure 2.1. Since the rate of load application is slow, QST does not account for inertial or damping forces in the structural system. Although not well-suited for simulating structural responses to earthquakes, QST is excellent for determining the hysteretic response of a member or structure. For this reason, QST results are often used to provide realistic

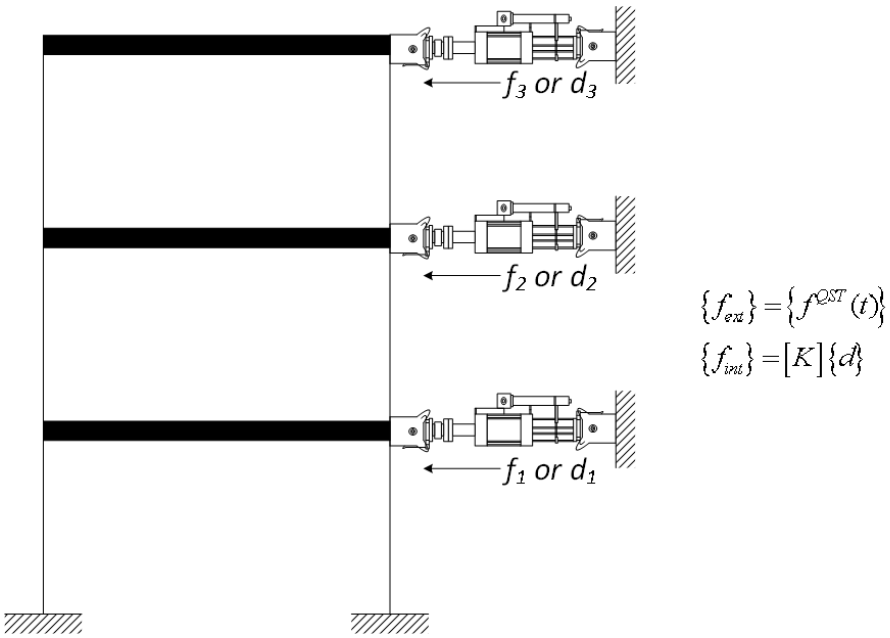



FIGURE 2.1: Quasi-static testing methodology

hysteresis models to enable more accurate analytical simulations of structural responses to seismic loading [Shao and Griffith, 2013].

For decades, STT has been widely used for simulation of seismic loading applied to scale-models of structures, as well as for qualification testing of structural components (Figure 2.2). STT is performed in real-time, so inertial and damping forces in the structure are naturally simulated [Shao and Enyart, 2014]. This is one of the greatest advantages of STT. However, due to the physical limitations of shake table hardware, scaling of full-scale structures is often unavoidable, since specimen size is limited by the payload and footprint of the available shake table. Unfortunately, scaling of the structure can present complications, as similitude is difficult to achieve without altering

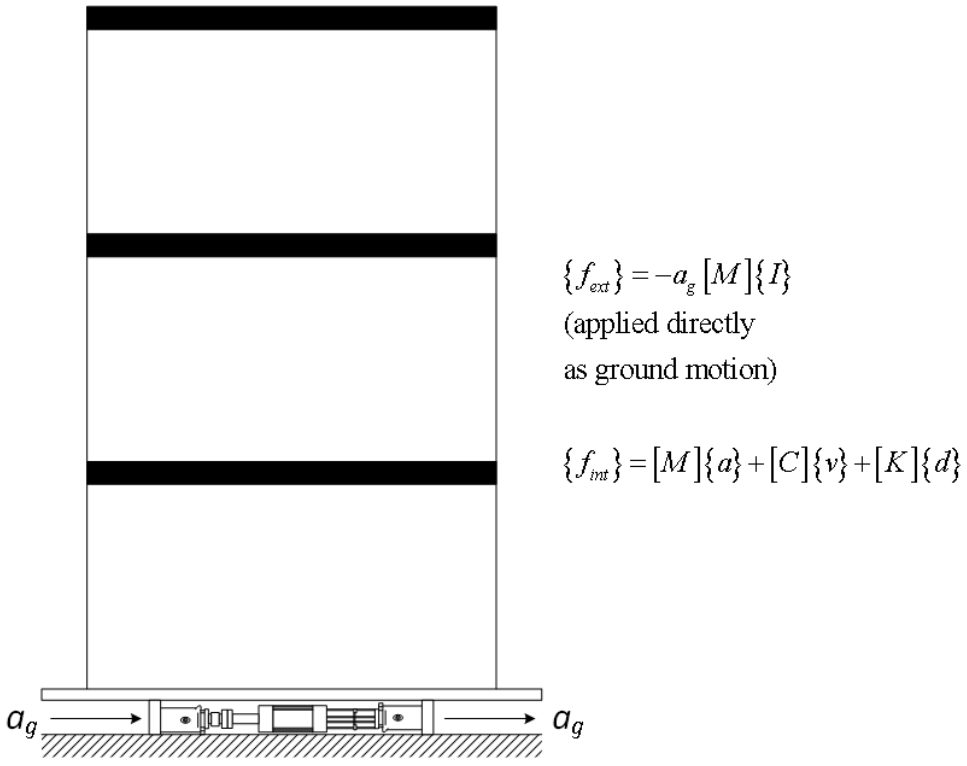



FIGURE 2.2: Shake table testing methodology

the dynamic properties of the structure [Mahin and Shing, 1985, Harris and Sabnis, 2010]. The available degrees of freedom for prescribed excitation may also be limited by the capabilities of the shake table.

EFT most closely parallels STT [Dimig et al., 1999], but it simulates ground motion indirectly through application of effective earthquake forces at discrete degrees of freedom of the structure rather than through direct base excitation (Figure 2.3). EFT is a force-control method in which the ground motion accelerations that would be applied by a shake table are converted into a set of effective forces using the same principles used for computational analysis of relative motion due to base excitation.

Prior to the test, the effective forces are computed as the product of the ground accelerations and the masses lumped at the degrees of freedom and are applied in the direction opposite the ground motion. Throughout the test, those predetermined

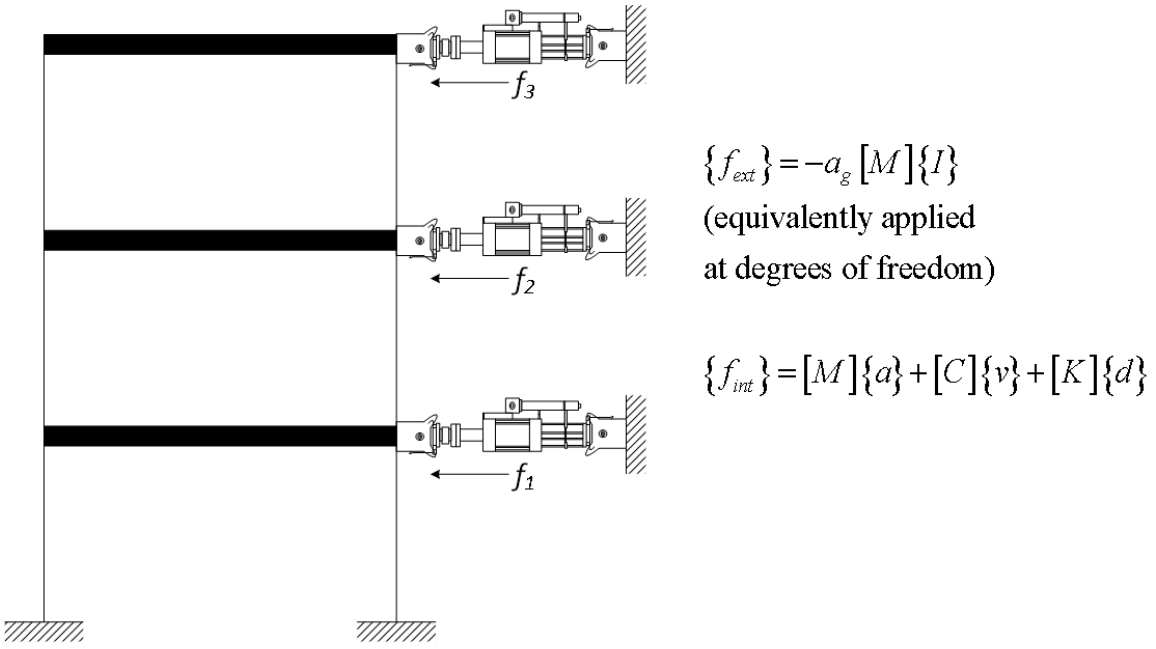



FIGURE 2.3: Effective force testing methodology

forces are applied to the structure at free degrees of freedom using hydraulic actuators, while the boundaries of the structure are restrained by a strong wall or strong floor, depending on the application. EFT is performed in real-time, so the inertial and damping forces developed in the structure are naturally accounted for, as in a STT. The advantage of EFT over STT is that the structure often need not be downscaled. Originally, the most notable obstacle to EFT was control-structure interaction, particularly natural velocity feedback of the actuator, but this issue has been largely resolved [Dyke et al., 1995, Dimig et al., 1999]. One other drawback is that the prescribed effective forces are based on a lumped-mass model assumption–an idealization not necessary for STT.

PSD, illustrated in Figure 2.4, is a displacement-control approach that combines the simplicity of QST with the utility of EFT using computational corrections for unbalanced forces. Its setup is very similar to that of QST, but the loading is determined

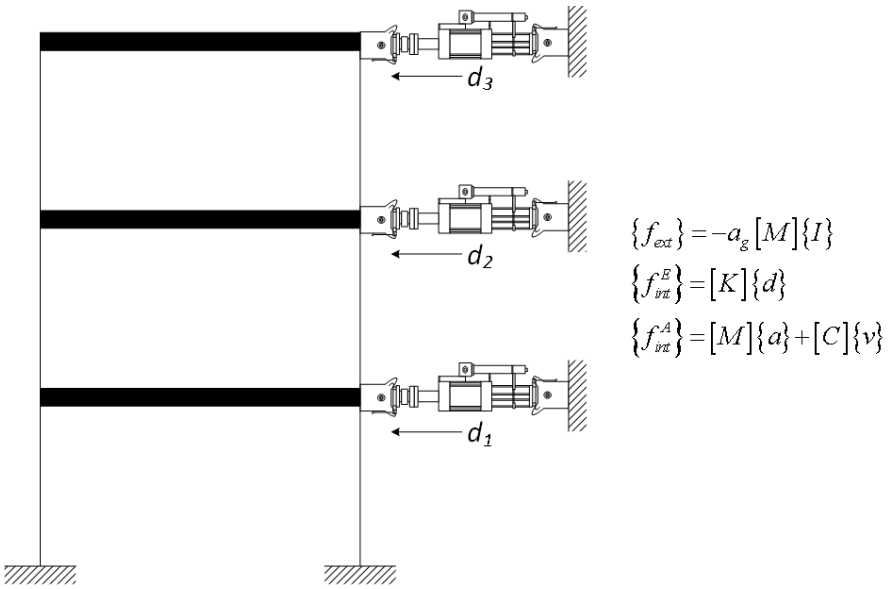



FIGURE 2.4: Slow pseudodynamic testing methodology

in a similar manner to that of EFT. In both EFT and PSD, the externally-applied loads are predetermined, but they are not imposed on the structure explicitly in PSD. Instead, the predetermined externally-applied loads are substituted into the equation of motion, and displacements predicted by time-stepping with a numerical integration scheme are applied to the structure [Mahin and Shing, 1985]. The true corresponding externally-applied forces for the prescribed displacements are then measured by load cells at the actuators. This method has been extended to hybrid testing approaches, including the hybrid simulation framework (HSF) presented in this thesis.

PSD can be performed either at a slow rate (as in QST) or in real-time (as in EFT), depending on the hardware and control capabilities of the test site. In the traditional slow PSD approach, inertial and damping forces within the structure are accounted for analytically through the equation of motion with assumed mass and damping matrices. Only the stiffness effects are evaluated experimentally. This is indicated by the separation of the internal force vector into experimental and analytical contributions, as denoted by the $E$ and $A$ superscripts in Figure 2.4. The main advantage of real-time over slow PSD is the ability to account for the dynamic effects of mass and damping directly. This advantage is particularly relevant when there is significant velocity-dependent behavior [Shao et al., 2010]. In cases where behavior of the structure is primarily independent of velocity, slow PSD is adequate, as real-time tests can be fairly complicated, require special compensation of actuator dynamics, and generally demand extensive validation, calibration, and laboratory resources [Shao and Griffith, 2013].

## 2.2 Overview of Hybrid Simulation

Simulations of structural responses to dynamic excitation have historically fallen into two categories: numerical or analytical simulations based on the finite element method and experimental testing. In the past few decades, researchers have been developing methods of simulation capable of merging the convenience and cost-efficiency of analytical simulation with the exploratory power and fidelity of experimental testing. Such mixed analytical/experimental strategies are known as hybrid simulations. Hybrid simulation involves both analytical and experimental simulation running in tandem to account for the response of complementary portions of a structure. Analytical simulation is performed on the portion of the structure where the behavior is fairly well-characterized and predictable under the applied loading, since analytical simulation on that substructure would be sufficient for the purposes of the investigation. Experimental testing is limited to only a substructure whose behavior cannot be reliably predicted with analytical methods, for one reason or another. Experimental testing may be preferred for a variety of situations, including:

- Complex material properties, as in the case of composite materials [Udagawa and Mimura, 1991, Gencturk and Elnashai, 2014]

- A critical substructure, such as a fuse element or other member expected to have high demand under specified loading and exhibit fracture, instability, or other failure mechanisms that may be challenging to faithfully model analytically [Shoraka et al., 2008]

- Complex geometries or interconnections between structural components that may be difficult to model analytically [Yang et al., 2006]

- Nonlinear behavior for which the existing mathematical model is incomplete [Mahmoud et al., 2013]

Development of hybrid simulation concepts originated in the mid-1970s. In its earliest form, hybrid testing was simply the basic PSD proposed by Takanashi [Takanashi et al., 1975] and refined by Shing, Mahin, and others throughout the 1980s [Mahin and Shing, 1985, Mahin et al., 1989]. Basic PSD is intrinsically a hybrid simulation method, since it involves numerical consideration of inertial and damping forces but experimental evaluation of structural stiffness. Even so, due to the fact that basic PSD requires a full structural model, it does not fully leverage the benefits of modern hybrid testing methodology that enable dynamically-consistent testing of individual structural components or substructures. Over the past few decades, principles of analytical dynamic substructuring have been applied to the original PSD formulation and its variations, resulting in the current family of hybrid simulation forms.

Most hybrid simulations to date have been based on extension of one of the following experimental simulation methods: slow or real-time PSD, real-time STT, real-time EFT, or combined real-time STT and EFT. For the HSF presented in this paper, slow pseudodynamic hybrid simulation (PSDHS) will be used. The general framework for this approach is illustrated in Figure 2.5, where the components in gray signify the analytical portion of the structural model. In PSDHS, the discrete-time dynamic equation of motion (Equation 2.1) is used to determine displacements to be applied

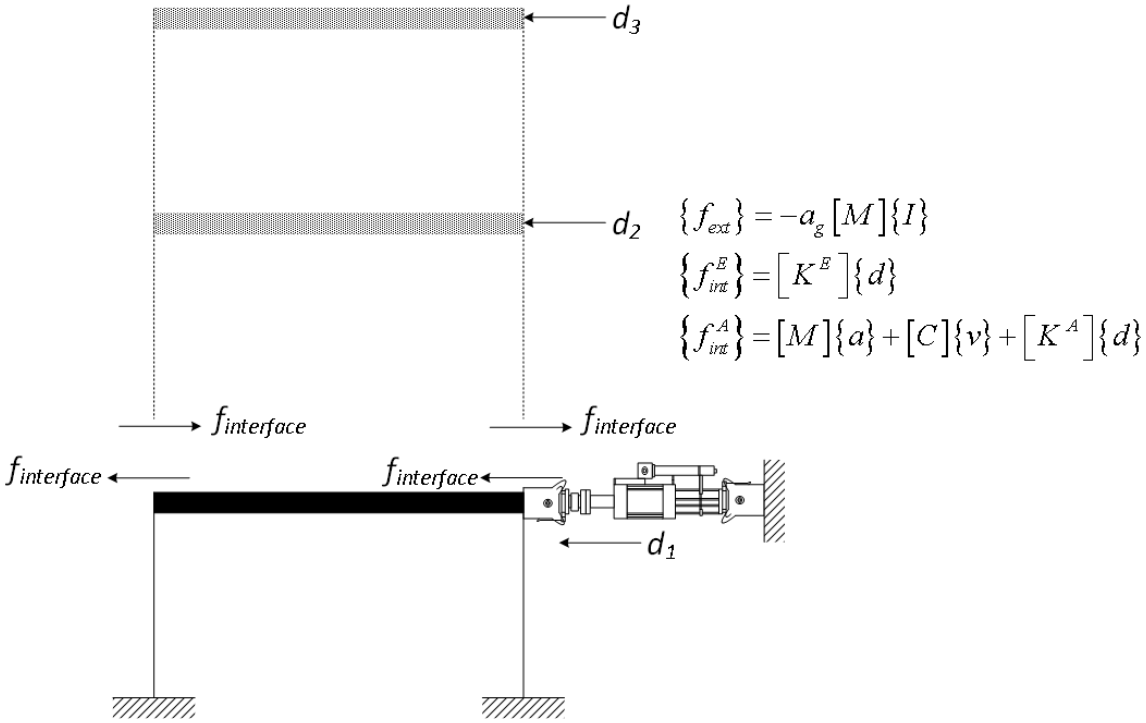



FIGURE 2.5: Slow pseudodynamic hybrid simulation methodology

to the structure at the next time step. In Equation 2.1, the **Kx** term may be taken as the restoring force vector **r**, composed of experimental and analytical parts, denoted by $E$ and $A$ superscripts. This formulation is expressed in Equation 2.2.

$$\mathbf{M}\ddot{\mathbf{x}}_n + \mathbf{C}\dot{\mathbf{x}}_n + \mathbf{r}_n^E + \mathbf{K}^A\mathbf{x}_n = \mathbf{f}_{ext} \tag{2.2}$$

For the experimental substructure, the restoring force due to the applied displacement is measured directly. For the analytical substructure, the restoring force is computed based on the stiffness of the analytical substructure and motion of the unrestrained degrees of freedom. If substitution of $\mathbf{r}^E$ into Equation 2.2 results in an imbalance between the left- and right-hand sides of the equation, the imbalance may be corrected using either a corrector step or an iterative method not involving physical application

of displacement corrections, since imposing physical displacement corrections may result in unintended permanent deformation of the specimen. In Figure 2.5, the interface forces are those that result in dynamic equilibrium for each substructure. The imbalance between the left- and right-hand sides of Equation 2.2 may be considered as the difference between the interface forces shown in Figure 2.5.

While real-time tests are preferred where practical, they are highly complex and often expensive to implement. In tests involving inelastic structural response, accurate application of a desired load or force on the structure is crucial, since the structural response is path-dependent. Accurate application of a hydraulic actuator load or displacement is much simpler in a slow test, where the load or displacement can be applied gradually, ramping up or down to the target value, thereby minimizing overshoot. In any type of real-time testing, ramping is difficult due to time constraints. Ramping to the target is especially inconvenient in the case of real-time PSD, which requires time for the computation of target displacements at each time step, in addition to the time required to actually apply the displacements experimentally. The complexity increases yet again with real-time PSDHS, since an analytical substructure must also be analyzed separately within each time step. For configurations with multiple degrees of freedom in the experimental substructure, multiple actuators may be required to work in unison, further adding to the complexity. While a number of laboratories have developed real-time capabilities with some success, slow PSDHS with one experimental degree of freedom will be the starting point for the HSF presented here. Slow PSDHS has the added benefit of allowing for an iterative integration scheme without the concern for the time-delay issues on the experimental side. This will be

important when considering geometric nonlinearity in the analytical substructure. In the future, improvements and upgrades may be made to the HSF as laboratory hardware resources become more capable.

Although hybrid simulation at this time is still predominantly of the slow PSD variety, two of the most versatile systems recently under development are real-time dynamic hybrid simulation (RTDHS) and real-time hybrid simulation (RTHS) [Shao et al., 2010]. RTDHS utilizes both shake tables and actuators. One or more shake tables apply ground motion, while actuators apply interface forces at the boundaries between the analytical substructure and the experimental substructure. RTHS is similar, but it "[allows] a redistribution of dynamic load between shake tables and dynamic actuators to adapt to the available laboratory equipment" [Shao et al., 2010].

## 2.3 Applications of Hybrid Simulation in Structural Testing

Hybrid simulation has been applied to a variety of structures for analysis of different applied loads. Test structures have included many steel frames, both braced and moment-resisting, as well as reinforced-concrete, masonry, and wood structures [Shao and Griffith, 2013]. Multi-span bridges have also been studied in several hybrid simulation projects [Frankie et al., 2013], with loads including both ground motion and vehicular traffic [Terzic and Stojadinovic, 2013]. In addition to ground motion and vehicular excitation, some researchers have performed hybrid simulation of gravity load collapses of reinforced-concrete frames [Shoraka et al., 2008]. Many of the applications of hybrid simulation have involved structures such as shear buildings that can be modeled using a few lateral degrees of freedom at horizontal diaphragms. Others have

considered multiple degrees of freedom on a single experimental substructure, such as the bracing in a steel zipper frame [Shing et al., 2006]. One of the most intriguing uses of hybrid testing is geographically-distributed simulation, where multiple substructures from the same structural model are evaluated in a single simulation utilizing a network of several test sites, such as the Network for Earthquake Engineering [Shao and Griffith, 2013]. Hybrid testing has also been used to test and validate nondestructive evaluation techniques and sensor technologies for structural health monitoring applications [Mercado and Zhang, 2012].

Despite the increased popularity of hybrid testing, the overwhelming majority of studies have been related to earthquake engineering research. For all the research that has been done, there is a surprising lack of studies applying hybrid simulation to lattice or truss structures, such as power transmission towers and telecommunications towers. As mentioned in the previous chapter, the focus application area of the HSF developed, verified, and validated in this thesis will be these systems.

Over the years, failures of power transmission towers have caused large, costly power outages and required expensive repairs [Klinger et al., 2011]. Most catastrophic transmission tower failures have occurred in extreme weather conditions, such as ice storms and wind storms. Ice storms with strong winds generate dynamic loading scenarios involving galloping of conductor wires [Baenziger et al., 1994], but literature review suggests that very few dynamic or pseudodynamic experiments have been performed on transmission towers to examine structural responses under these loads. Failure analysis over the past few decades has often employed full-scale testing [Albermani et al., 2009, Rao et al., 2010, Rao et al., 2012, Fan et al., 2009, Landers, 1982],

but due to the prohibitive costs of full-scale testing of these towers, researchers have developed nonlinear finite element analysis procedures to predict failure of many tower designs under static loads comparable to those applied in full-scale tests [Eltaly et al., 2014]. Dynamic analysis by the finite element method is a powerful tool [Shen et al., 2010], but experimental methods provide the only means to validate the reliability of computational dynamic analysis. Critical components of power transmission systems are challenging to analyze, particularly in regard to failure analysis considering material and geometric nonlinear behavior of the structural members. As pointed out by Fan in a paper for the 2009 ASCE Electrical Transmission and Substation Structures Conference, "Full scale tower tests provide an indispensable tool to validate the adequacy of the structures designed" [Fan et al., 2009].

Admittedly, construction and dynamic loading of entire full-scale towers, not to mention a multi-span system, is often impractical. Hybrid testing could serve as an effective alternative by executing pseudodynamic simulations of full-scale components of a system or tower while accurately accounting for the interaction with the rest of the structural system. Such testing has the potential to provide a compromise between the reliability of full-scale testing and the convenience and cost advantages of computer simulation, perhaps in a way that is similar to scale modeling as presented by Richardson [Richardson, 1987], but serving to answer questions of detail rather than overall structural behavior. If so, hybrid simulation could play a role in improving the quality of the design of new towers as well as identifying and correcting weaknesses of specific existing tower designs that have failed in the field. Additionally, a recent state-of-the-art review [Chen et al., 2014] has identified several needs for greater

understanding of transmission tower-line system response to various dynamic loadings that are not well-modeled analytically at the present time. Hybrid testing could make research in those areas more economically feasible and expedient.

# CHAPTER 3: THEORETICAL FOUNDATION

In order to achieve a macroscopic understanding of the HSF that has been developed, the underlying theory must first be introduced. Geometric nonlinear analysis concepts discussed by McGuire et al. [McGuire et al., 2000] are implemented within the HSF, with particular treatment of geometric nonlinearity of structures comprised of axial elements. The direct time integration procedure used in the HSF is based primarily on the $\alpha$-operator splitting integration scheme presented by Combescure and Pegon [Combescure and Pegon, 1997] and the iterative method devised by Mosqueda and Ahmadizadeh for dealing with logistical challenges associated with nonlinear iterations on the experimental substructure [Mosqueda and Ahmadizadeh, 2011]. Each of these topics is addressed in this chapter.

## 3.1 Nonlinear Geometric Effects

As a structure undergoes displacements due to applied loads, changes in geometry affect the load path. In many structural analysis applications, displacements are small enough to allow the analyst to neglect the change in geometry of a system without introducing significant error into the analysis. In applications with larger displacements, the geometry of the displaced configuration must be taken into account in the formation of the stiffness matrix for the system. Such analysis is classified as second-order (or large displacement) analysis, and structural response determined by

second-order analysis is nonlinear, even with purely linear-elastic material behavior. For axial elements, the incorporation of these geometric nonlinear effects within the stiffness matrix is a fairly simple task, but the solution of the equilibrium equations requires an iterative method. For a single axial element with nonlinear geometric effects, linear elastic material properties, and degrees of freedom as indicated in Figure 3.1, the elastic, geometric, and tangent stiffness matrices in local coordinates are

$$\mathbf{k}_e = \frac{EA}{L}\begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} u_1 \\ v_1 \\ w_1 \\ u_2 \\ v_2 \\ w_2 \end{matrix} \qquad \mathbf{k}_g = \frac{f_x}{L}\begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} u_1 \\ v_1 \\ w_1 \\ u_2 \\ v_2 \\ w_2 \end{matrix}$$

$$\mathbf{k} = \mathbf{k}_e + \mathbf{k}_g \tag{3.1}$$

where $L$ and $f_x$ are member length in the reference configuration and internal member force, respectively.

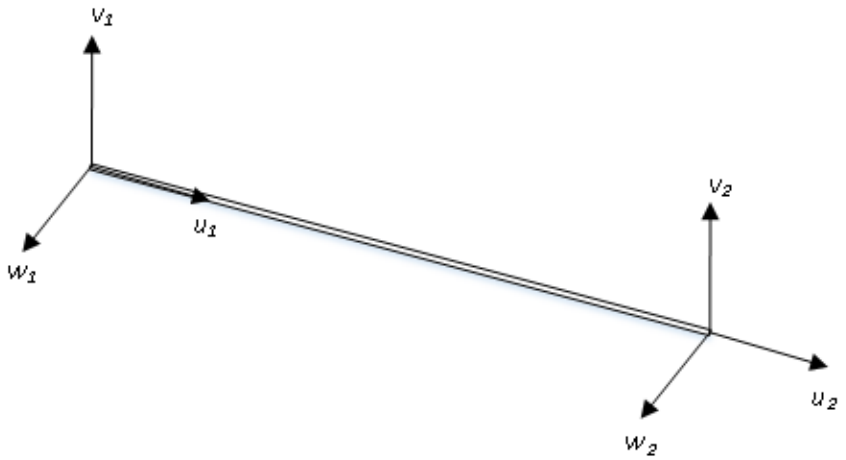



FIGURE 3.1: Local degrees of freedom for elements in example model

In Equation 3.1, $E$ and $A$ are Young's modulus and the cross-sectional area of the member, respectively. Throughout this thesis, $A$ for a given member is assumed to be constant (i.e. Poisson's effect is neglected). The global tangent stiffness matrix $\mathbf{K}$ is formed by computing the local tangent stiffness matrix $\mathbf{k}$ for each element, transforming each local stiffness matrix to global coordinates, and summing the contributions of all the individual element stiffnesses by the direct stiffness method. For an increment of relative displacement applied to an axial element (neglecting rigid body motion), the force in the element at the end of the increment can be simply determined using the stiffness coefficient of the element at the beginning of the increment as

$$f_{x,n} = \frac{EA + f_{x,n-1}}{L_{n-1}}(L_n - L_{n-1}) + f_{x,n-1} \tag{3.2}$$

This equation can be rearranged to

$$\frac{f_{x,n} - f_{x,n-1}}{L_n - L_{n-1}} = \frac{EA + f_{x,n-1}}{L_{n-1}} \tag{3.3}$$

which, assuming the material properties are constant within the increment, indicates that the change in member force with respect to change in member length for an axial element is equal to its stiffness coefficient in the reference configuration. This means that in the absence of material nonlinearity, the local axial stiffness remains linear. The nonlinearity introduced into such a truss system is primarily due to global geometric changes, which can largely be accounted for by continuous updating of member transformation matrices.

To demonstrate the concept of nonlinear geometry in a truss, consider the two-element truss shown in Figure 3.2, which is based on Example 9.1 of *Matrix Structural*

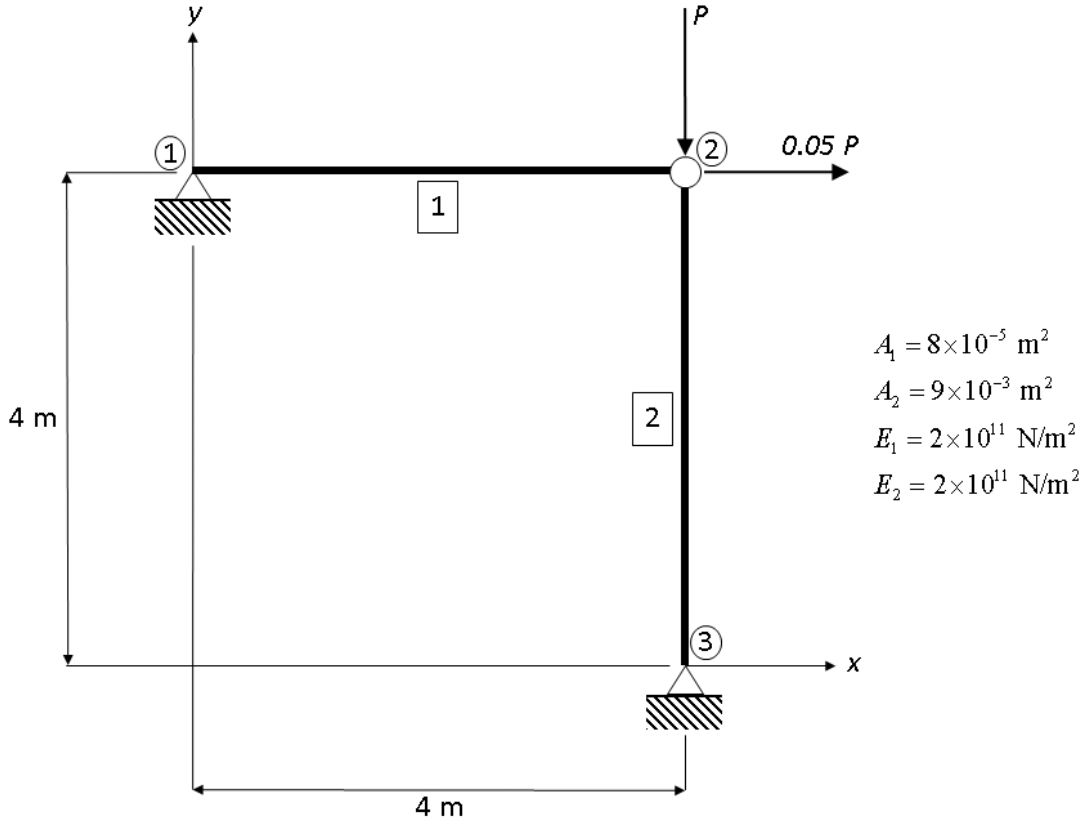



FIGURE 3.2: Two-degree-of-freedom truss model used for geometrically nonlinear analysis

*Analysis* [McGuire et al., 2000]. Following Equation 3.1, for the initial increment of a geometrically nonlinear analysis, the stiffness matrices of the two elements (with entries associated with the out-of-plane direction partitioned and removed) are

$$\mathbf{k}_1 = \frac{(2\times10^{11})(8\times10^{-5})}{4}\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \frac{f_x^{(1)}}{4}\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \text{[N/m]}$$

$$\mathbf{k}_2 = \frac{(2\times10^{11})(9\times10^{-3})}{4}\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \frac{f_x^{(2)}}{4}\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \text{[N/m]}$$

Using transformation matrices, each of the $n$ local element stiffness matrices is transformed from local to global coordinates. The contribution of each element to the global stiffness matrix can be formulated from the element local stiffness matrix and corresponding transformation matrix as

$$\mathbf{K}_n = \mathbf{T}_n^T \mathbf{k}_n \mathbf{T}_n \tag{3.4}$$

For any axial element, the transformation matrix, $\mathbf{T}$, is given by Equation 3.5:

$$\mathbf{T} = \begin{bmatrix} l & m & n & 0 & 0 & 0 \\ -\frac{m}{D} & \frac{l}{D} & 0 & 0 & 0 & 0 \\ -\frac{ln}{D} & -\frac{mn}{D} & D & 0 & 0 & 0 \\ 0 & 0 & 0 & l & m & n \\ 0 & 0 & 0 & -\frac{m}{D} & \frac{l}{D} & 0 \\ 0 & 0 & 0 & -\frac{ln}{D} & -\frac{mn}{D} & D \end{bmatrix} \tag{3.5}$$

where

$$l = \frac{x_2 - x_1}{L} \qquad m = \frac{y_2 - y_1}{L} \qquad n = \frac{z_2 - z_1}{L} \qquad D = \sqrt{l^2 + m^2}$$

The above formulation for the transformation matrix follows Section 5.5 of Logan's finite element text [Logan, 2011]. Since this example is planar in the $x$-$y$ plane, the

third and sixth columns and rows of $\mathbf{T}$ can be omitted. Thus, for the configuration corresponding to the initial increment, the transformation matrices for the two elements in the example structure are constructed as

$$\mathbf{T_1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{T_2} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

After assembling the global stiffness matrix by the direct stiffness method and partitioning and removing all entries associated with restrained degrees of freedom, the global stiffness matrix $\mathbf{K}$ for the initial increment is

$$\mathbf{K} = 1 \times 10^6 \begin{bmatrix} 4 & 0 \\ 0 & 450 \end{bmatrix} + f_x^{(1)} \begin{bmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{4} \end{bmatrix} + f_x^{(2)} \begin{bmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{4} \end{bmatrix} \text{[N/m]}$$

Figure 3.3 shows the result of the nonlinear static analysis for this example, solving iteratively for $P$ in $du = 1$ mm increments of horizontal displacement of Node 2 from $u = 0$ m to $u = 3$ m. The dashed curve shows the results that are obtained by first-order linear analysis. The solid curve shows nonlinear geometric analysis results, where the stiffnesses and transformation matrices have been updated at each displacement increment to account for the geometric nonlinearity present in the system. Again, it should be noted that changes in the cross-sectional areas of the elements due to Poisson's effect are neglected here; only axial strain is considered.

For small displacements, the initial stiffness is approximately equal to the updated tangent stiffness. In the example above, "small displacement" might be defined as

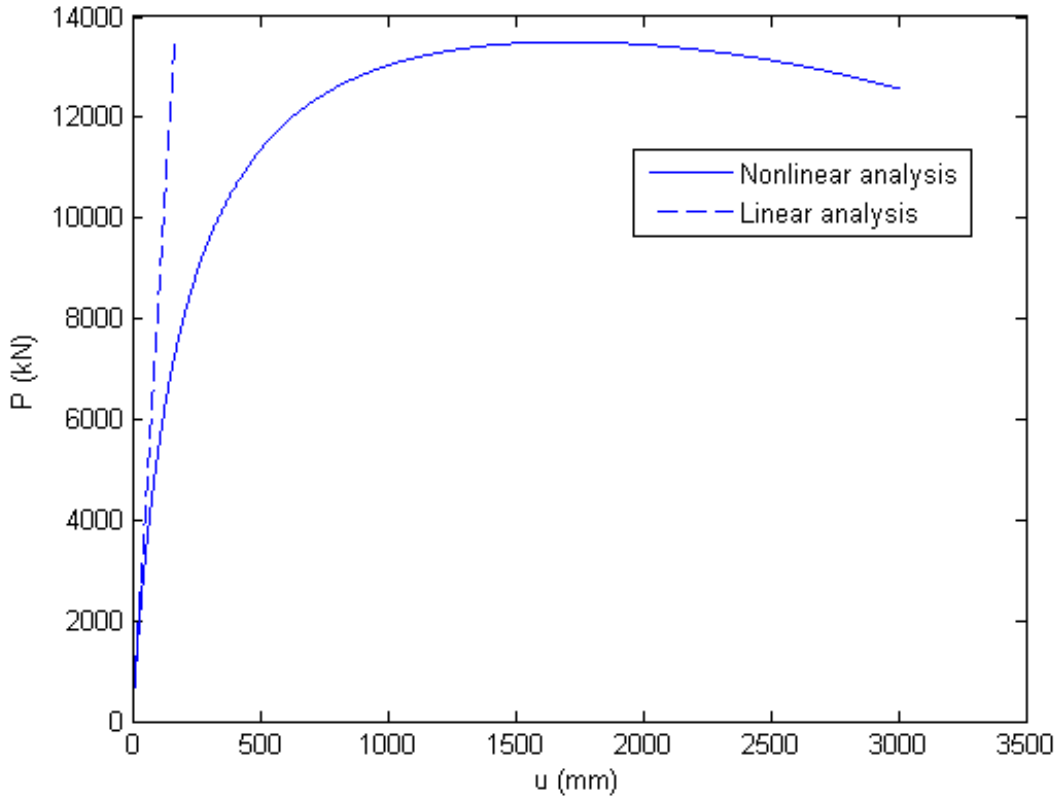



FIGURE 3.3: Load-displacement curves for static analysis of structure in Figure 3.2

$u \leq 50$ mm. For larger displacements, such an approximation is clearly inaccurate, and geometric nonlinearity becomes significant. Figure 3.4 shows the dramatic difference between the original and limit-point configurations. It is apparent that the original transformation matrices are significantly inaccurate at this stage of nonlinearity. In many approaches to nonlinear geometric analysis, the reference configuration for each increment remains constant as the member forces are updated. In the HSF presented in this thesis, the reference configuration is updated at each iteration, since the displacements for the entire structure are projected at each iteration.

For elements subject to bending stresses, geometric nonlinearity becomes nontrivial

on the local level as well and must be appropriately considered. End rotations may generate significant geometric effects in a moment-resisting member, even with relatively small translational displacements of the ends of the member. Further discussion of the topic of geometric nonlinearity in flexural members is beyond the scope of this thesis. The reader is referred to Chapter 9 of *Matrix Stuctural Analysis* by McGuire et al. for additional discussion of geometric nonlinear analysis, including derivation of the geometric stiffness matrix for an axial element. Appendix B of the same text also provides insight into nuanced consideration of the effects of rigid body motion and natural deformation in large-displacement problems, as well as derivation of Equation 3.2.

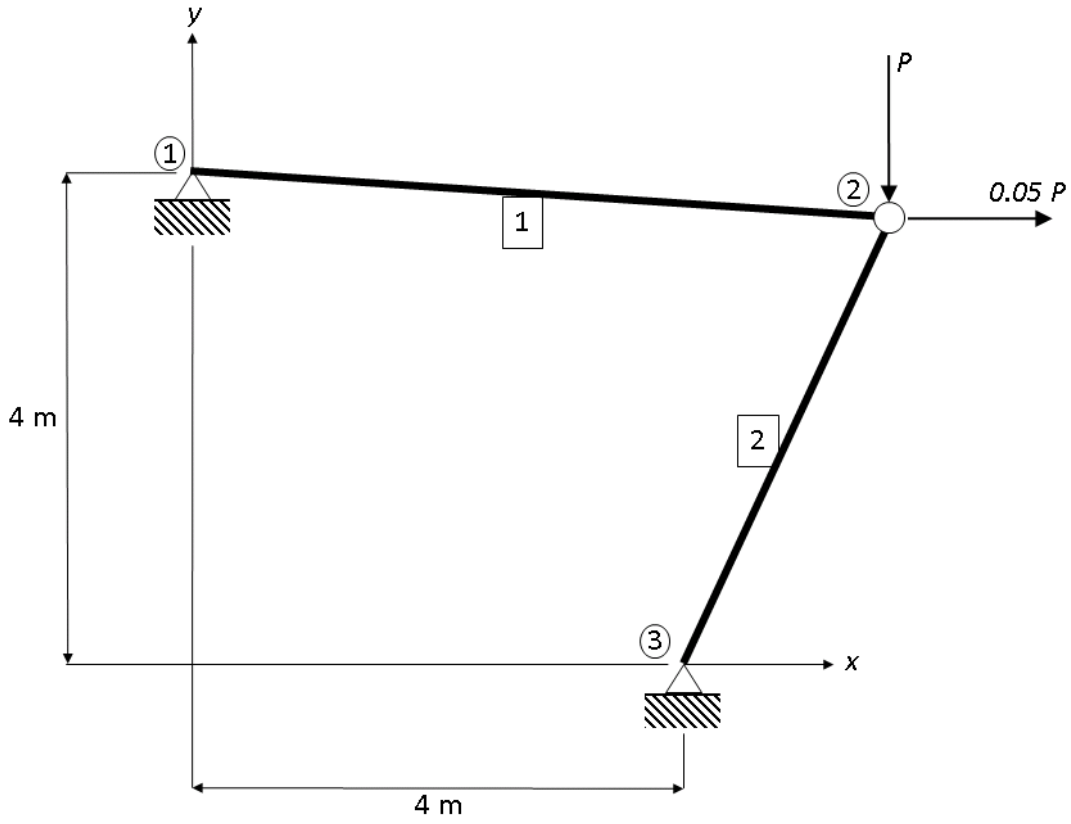



FIGURE 3.4: Limit-point deformed geometry for nonlinear static analysis

## 3.2 $\alpha$-Operator Splitting Integration Scheme

The $\alpha$-operator splitting ($\alpha$-OS) method of time integration as explained by Combescure and Pegon [Combescure and Pegon, 1997] is the basis for the integration scheme presented in this thesis. Since the procedure outlined by Combescure and Pegon was tailored to PSD, analytical and experimental substructuring is not addressed in this section. Furthermore, the determination of restoring forces described here is only applicable to the experimental substructure of a hybrid simulation and must be complemented by a comparable method of computing restoring forces for the analytical substructure in order to be implemented satisfactorily in hybrid applications.

### 3.2.1 Discrete-Time Equation of Motion

The $\alpha$-OS method is a modification of the Hilber-Hughes-Taylor (HHT) method widely used in most commercial finite element analysis packages for direct time integration [Hilber et al., 1977]. The HHT method is a special one-parameter ($\alpha$) method that is a subclass of the three-parameter ($\alpha$, $\beta$, $\gamma$) family of extended Newmark integration methods. In the HHT method, $\beta$ and $\gamma$ are expressed as functions of $\alpha$:

$$\beta = \frac{(1-\alpha)^2}{4} \tag{3.6}$$

$$\gamma = \frac{1-2\alpha}{2} \tag{3.7}$$

where $\alpha \in [-\frac{1}{3}, 0]$. In discrete-time integration, these three parameters are used to compute the displacement and velocity vectors at the next time step in terms of the acceleration vector of the next time step and the displacement, velocity, and

acceleration vectors of the current time step:

$$\mathbf{d}_n = \mathbf{d}_{n-1} + \Delta t \mathbf{v}_{n-1} + \frac{\Delta t^2}{2}(1-2\beta)\mathbf{a}_{n-1} + \Delta t^2 \beta \mathbf{a}_n \tag{3.8}$$

$$\mathbf{v}_n = \mathbf{v}_{n-1} + \Delta t(1-\gamma)\mathbf{a}_{n-1} + \Delta t \gamma \mathbf{a}_n \tag{3.9}$$

where $\mathbf{a}_n$ is unknown. By substitution into the discrete-time equation of motion (Equation 2.1), the state vectors at the next time step are related through the following equation:

$$\mathbf{M}\mathbf{a}_n + (1+\alpha)\mathbf{C}\mathbf{v}_n - \alpha\mathbf{C}\mathbf{v}_{n-1} + (1+\alpha)\mathbf{r}_n - \alpha\mathbf{r}_{n-1} = (1+\alpha)\mathbf{f}_n - \alpha\mathbf{f}_{n-1} \tag{3.10}$$

where $\mathbf{r}$ and $\mathbf{f}$ are the vectors of restoring force and external applied force, respectively.

### 3.2.2 Solution of the Equation of Motion by a Predictor-Corrector Method

As pointed out by Combescure and Pegon [1997], the relationship expressed by Equation 3.10 is implicit, since the unknown $\mathbf{a}_n$ is used to produce $\mathbf{v}_n$ and $\mathbf{d}_n$. The restoring force vector, $\mathbf{r}_n$, is also dependent on $\mathbf{a}_n$, albeit indirectly, since the experimentally-measured $\mathbf{r}_n$ is a function of predicted displacement and the stiffness of the experimental substructure. To solve Equation 3.10 for $\mathbf{a}_n$, the $\alpha$-OS method uses a predictor step to explicitly predict $\mathbf{a}_n$ in Equation 3.10 by temporarily neglecting the $\mathbf{a}_n$ terms of Equations 3.8 and 3.9. Using this assumption of $\mathbf{a}_n$, a prediction of $\mathbf{d}_n$ is determined from Equation 3.8 and used to find an initial estimate of the restoring force vector as described in Section 3.2.3. The predictor displacement is denoted by $\tilde{\mathbf{d}}$, and the associated restoring force vector is denoted by $\tilde{\mathbf{r}}$. An initial prediction of the velocity vector, $\tilde{\mathbf{v}}$, is computed as well, using Equation 3.9. At this point, predicted

values for all quantities in Equation 3.10 are known, and Equation 3.10 may be solved for $\mathbf{a}_n$. To solve for $\mathbf{a}_n$, the pseudo-force vector, $\hat{\mathbf{f}}_n$, for the time step and pseudo-mass matrix, $\hat{\mathbf{M}}$, are computed as follows:

$$\hat{\mathbf{f}}_n = \mathbf{f}_n + \alpha\Delta\mathbf{f} - \mathbf{C}\tilde{\mathbf{v}}_n - \tilde{\mathbf{r}}_n - \alpha\left[\mathbf{C}\Delta\mathbf{v} + \Delta\mathbf{r} - \left(\mathbf{C}\Delta t\gamma + \mathbf{K}\Delta t^2\beta\right)\mathbf{a}_{n-1}\right] \tag{3.11}$$

and

$$\hat{\mathbf{M}} = \mathbf{M} + \mathbf{C}\Delta t\gamma + \mathbf{K}\Delta t^2\beta + \alpha\left[\mathbf{C}\Delta t\gamma + \mathbf{K}\Delta t^2\beta\right] \tag{3.12}$$

where $\mathbf{K}$ is the tangent stiffness matrix. The acceleration vector for the time step is computed as the solution of a system of linear equations:

$$\hat{\mathbf{M}}\mathbf{a}_n = \hat{\mathbf{f}}_n \tag{3.13}$$

Following solution of Equation 3.10 for $\mathbf{a}_n$, the displacement and velocity vectors for the next time step are corrected by reintroducing the terms associated with $\mathbf{a}_n$ that were initially neglected. This step of the procedure is commonly known as the corrector step. As an alternative to completely neglecting $\mathbf{a}_n$ terms in the predictor step as described by Combescure and Pegon, Ahmadizadeh [2007] suggests that the $\beta$ and $\gamma$ values for Equations 3.8 and 3.9 be temporarily set to zero to compute the predictor displacement and velocity. In that case, the corresponding corrected displacement and velocity are determined by reintroducing the terms involving $\beta$ and $\gamma$ following solution of Equation 3.10. This approach is adopted in the HSF presented in this thesis and described in Section 3.3.

### 3.2.3 Computation of the Restoring Force Vector

The restoring force vector in the equation of motion is determined through a three-step process. First, the measured restoring force, $\tilde{\mathbf{r}}^m$, is determined by direct measurement of the restoring force resulting from application of the desired displacement. Second, $\tilde{\mathbf{r}}^m$ is numerically adjusted to compensate for experimental errors. In order to compensate for experimental error associated with the discrepancy between the predictor displacement, $\tilde{\mathbf{d}}$, and the displacement actually applied, $\tilde{\mathbf{d}}^m$, the restoring force vector associated with $\tilde{\mathbf{d}}$ may be approximated as

$$\tilde{\mathbf{r}} = \tilde{\mathbf{r}}^m + \mathbf{K}\left(\tilde{\mathbf{d}} - \tilde{\mathbf{d}}^m\right) \tag{3.14}$$

where $\mathbf{K}$ is the tangent stiffness of the experimental substructure. This correction assumes that the errors in measurement are small in comparison to the error in command displacement. Usage of Equation 3.14 requires an analytical estimate of the stiffness matrix of the experimental substructure [Combescure and Pegon, 1997]. Following solution of the predictor step for $\mathbf{a}_n$ and computation of the corresponding $\mathbf{d}_n$, the corrected restoring force vector is determined by Equation 3.15:

$$\mathbf{r} = \tilde{\mathbf{r}} + \mathbf{K}\left(\mathbf{d} - \tilde{\mathbf{d}}\right) \tag{3.15}$$

where $\mathbf{K}$ is again the analytically-constructed tangent stiffness, which is assumed to be same for the predicted displacement and the corrected displacement.

## 3.3 Iterative Solution of the Equation of Motion in Hybrid Testing

For structures subject to nonlinearities, the tangent stiffness matrix **K** and the damping matrix **C** may vary with displacement and velocity. Where practical, incremental-iterative solution methods can be used to achieve more accurate solutions to these problems than simple predictor-corrector steps [Mosqueda and Ahmadizadeh, 2011]. As mentioned in Section 2.2, structural response is path-dependent in many cases. This principle is the reason overshoot of target displacements in PSD is so greatly discouraged. For the same reason, physical methods of iterating on the experimental substructure are generally thought to be unsatisfactory, as they may produce unintended irreversible effects (namely plastic deformations) in the specimen and compromise the fidelity of the simulation. The method proposed by Mosqueda and Ahmadizadeh provides an innovative way to iterate on a system with experimental substructures without imposing physical iterative displacements. The technique approximates the local force-displacement relationship for the experimental substructure through second-order polynomials fitted to the most recent data points of force-time and displacement-time plots for the physical measurements taken in the first iteration of each time step. For non-initial iterations in each time step, the physical imposition of iterative displacement on the experimental substructure is replaced with numerical iterations, using the approximation of the measured force-displacement relationship. This technique is adapted for the hybrid simulation framework presented in this thesis and will be discussed in detail in Section 4.3.

In a sense, the predictor-corrector method described in Section 3.2 can be categorized

as a special type of iterative method, where the number of iterations is fixed at 1. The iterative method used here, then, is simply an extension of that, where the maximum number of iterations per increment or time step is greater than 1. The inaccuracy of the $\alpha$-OS method arises from the fact that the stiffness and damping matrices obtained in the predictor step are only as accurate as the predicted state vectors upon which they are based. If the prediction is modified iteratively until the correction of the state vectors for the increment becomes negligible, the stiffness and damping matrices in hand at the end of the time step are much more accurate. The predictor-corrector method studied by Combescure and Pegon (see Section 3.2) is based on an initial assumption of zero *acceleration* for the time step, whereas the modified predictor-corrector proposed by Ahmadizadeh is based on an initial assumption of zero *change in acceleration* for the time step [Ahmadizadeh, 2007]. For nonlinear problems, Equations 3.8, 3.9, and 3.10 may be rearranged and solved iteratively with relative ease after initially assuming $\Delta\mathbf{a} = \{0\}$:

$$\mathbf{d}_n = \mathbf{d}_{n-1} + \Delta t \mathbf{v}_{n-1} + \frac{\Delta t^2}{2}\mathbf{a}_{n-1} + \Delta t^2 \beta \Delta\mathbf{a} \tag{3.16}$$

$$\mathbf{v}_n = \mathbf{v}_{n-1} + \Delta t \mathbf{a}_{n-1} + \Delta t \gamma \Delta\mathbf{a} \tag{3.17}$$

$$\mathbf{M}\mathbf{a}_n + \mathbf{C}\mathbf{v}_n + \mathbf{r}_n + \alpha\left[\mathbf{C}\Delta\mathbf{v} + \Delta\mathbf{r}\right] = \mathbf{f}_n + \alpha\Delta\mathbf{f} \tag{3.18}$$

where $\mathbf{r}_n$ is computed as the sum of experimental and analytical parts as shown in Equation 2.2 and $\Delta\mathbf{r}$ is computed as the change between $\mathbf{r}_{n-1}$ and $\mathbf{r}_n$.

Since the actual incremental acceleration, $\Delta\mathbf{a}$, is unknown and subject to iteration, the left-hand side of Equation 3.18 may not be balanced with the right-hand side. To

determine the amount by which to adjust $\Delta\mathbf{a}$, the unbalanced force, $\hat{\mathbf{f}}$, is computed as

$$\hat{\mathbf{f}} = \mathbf{f}_n + \alpha\Delta\mathbf{f} - \mathbf{M}\mathbf{a}_n - \mathbf{C}\mathbf{v}_n - \mathbf{r}_n - \alpha\left[\mathbf{C}\Delta\mathbf{v} + \Delta\mathbf{r}\right] \tag{3.19}$$

and pre-multiplied by the inverse of $\hat{\mathbf{M}}$, which is the sum of the matrices multiplied with $\Delta\mathbf{a}_n$ in each term of Equation 3.18:

$$\hat{\mathbf{M}} = \mathbf{M} + \mathbf{C}\Delta t\gamma + \mathbf{K}\Delta t^2\beta + \alpha\left[\mathbf{C}\Delta t\gamma + \mathbf{K}\Delta t^2\beta\right] \tag{3.20}$$

Note that the mass matrix is included in Equation 3.20, since it is multiplied by $\mathbf{a}_n$ in Equation 3.18, where $\mathbf{a}_n = \mathbf{a}_{n-1} + \Delta\mathbf{a}$. The rationale for including the damping and stiffness terms in Equation 3.20 can be traced back through Equation 3.18 and Equations 3.16 and 3.17, keeping in mind that the incremental restoring force vector $\Delta\mathbf{r}$ is computed by a secant stiffness for the increment multiplied by the incremental displacement.

Finally, the incremental acceleration is adjusted using

$$\Delta\mathbf{a}_n^{j+1} = \Delta\mathbf{a}_n^j + \hat{\mathbf{M}}^{-1}\hat{\mathbf{f}} \tag{3.21}$$

where $n$ and $j$ indicate the time step number and the iteration number, respectively. The process is repeated as needed in order to satisfy the equation of motion (Equation 3.18) within some desired convergence tolerance. The stiffness and damping matrices are updated at the start of each iteration based on the state vectors produced by the previous iteration. The stiffness is first recalculated according to the method described in Section 3.1. Then, the damping matrix is updated using the new stiffness matrix, the original mass matrix, and the assigned proportional damping coefficients. For this

thesis, stiffness and damping (since damping is dependent on stiffness) are treated as only displacement-dependent, so only $\mathbf{d}_n$ is needed to adjust $\mathbf{K}$ and $\mathbf{C}$. Additional aspects of the hybrid simulation technique and specifics on the implementation of the general iterative integration scheme are presented in detail within the following chapter, which focuses on the development of the HSF.

# CHAPTER 4: DEVELOPMENT OF A HYBRID TESTING FRAMEWORK FOR LATTICED STRUCTURES WITH NONLINEAR GEOMETRIC EFFECTS

In developing the HSF, a library of MATLAB functions was created to perform the dynamic analysis and interface with the experimental hardware. Some functions serve multiple purposes and are called from several levels of the main routine, while others are used to modularize certain subroutines of the HSF software, so that similar functions may be readily interchanged in future upgrades to the HSF. In this chapter, the main HSF software routine is presented along with discussion of its constituent functions. The HSF software routine can be described as a three-tier process consisting of (1) the user interface and definition of problem-specific inputs, (2) the HSF integration scheme with nonlinear equilibrium iterations, and (3) the experimental substructure operations performed within the integration scheme to incorporate the physical test specimen into the dynamic analysis. These three tiers are presented in algorithm form in this chapter. Source codes for technical aspects of the HSF are included in the appendices.

## 4.1 User Interface

The first tier of the HSF software routine is comprised primarily of user interface functions and a series of interactive, guided steps for defining problem-specific inputs. In this section, the function that provides the primary user interface for the HSF software is outlined, along with some of the responsibilities of the user for generating

meaningful results. The function *HybridSimulation.m* (Appendix A) is designed to guide the user through this setup process by providing analysis of fundamental dynamic properties of the model, the results of which inform the decisions of the user in establishing parameters for the direct-time integration scheme. The process encompassed by this high-level user interface routine is presented in Algorithm 1 and described in the following subsection.

*HybridSimulation.m* contains graphical user interface prompts for most of the necessary inputs from a user. The remaining required inputs are provided as arguments to the function. Upon executing the main routine, dialog boxes prompt the user to select an input file for the truss setup (geometry, material properties, restraints, etc.). If the truss input file is a text file, the formatting is required to be exactly as shown in Appendix B, with tab delimiters between entries in each row of the file, though the ordering of the sections of the file is unimportant. Alternatively, a spreadsheet may be used. Truss input files should be in consistent units internally. The truss input file

**Algorithm 1** User Interface Procedure

**Inputs:** samping rate ($Fs$), maximum number of iterations ($jlimit$), relative convergence tolerance ($tol$), numerical damping parameter ($\alpha$)

**Outputs (to results folder):** figures, results, status of each time step

1: Get input file for truss setup
2: Read input file and set associated units
3: Load excitation files and set associated units and scale factors
4: Build **M** and **K** for the initial linear elastic model
5: Perform modal analysis for natural frequencies, $\{\omega\}$, and vibration modes, $[\Phi]$
6: Check that $\omega_1$ coincides with a significant excitation content in the frequency spectrum
7: Specify proportional damping constants directly or by damping ratio-frequency pairs
8: **do** *IntegScheme.m* (Algorithm 2)
9: Plot and save results

is read and used to create a set of eight global variables: `UndeformedNodes`, `nnodes`, `Elements`, `nelements`, `E`, `A`, `Rho`, and `FreeDOFs`. The first four store the geometry and connectivity of the structure, while the latter four store the material and section properties, as well as information needed for matrix partitioning of the restrained degrees of freedom in the structure.

Following the loading of the truss input files, the user is prompted to specify the member of the structure that is to be represented with the physical test specimen and nodes desired for displacement outputs. Note that the results are saved in a text file for all nodes, but only the displacements of the selected nodes are plotted. Specimen and node selections are made in the functions *SpecimenSelect.m* and *NodeSelect.m*, which develop a plot of the model geometry and prompt the user to select by clicking the desired items. Selected items are highlighted in green for easy identification. Items may be deselected by a second click. For three-dimensional structures, the user may wish to rotate about the plot of the truss for a better view and access to members on the far side of the truss.

After the truss and experimental substructure (specimen member) are defined, dialog boxes prompt the user to select excitation time history and/or ground acceleration time history files to be loaded. These input files are required to be in the same format as shown in Appendix C, with a column for time and columns for $x$, $y$, and $z$ directions. Following the selection of each excitation file, the excitation is read, re-sampled, and scaled in magnitude according to the selections of the user. The restructured excitation is stored within the structure `Excitations`, which contains information regarding each excitation applied. The variable `Fexcite` contains the

effective force vectors for the total effect of all the excitations specified for each time step of the simulation.

Following this definition of the model, the lumped-mass matrix and the initial linear elastic stiffness matrix are assembled. The function used to construct the initial linear elastic stiffness matrix (*KTotal.m*, provided in Appendix D) is used throughout the HSF software routine for construction of stiffness matrices. Using the mass and stiffness matrices, the generalized eigenvalue problem is solved, yielding estimates of the linear normal modes and corresponding natural frequencies of the initial model. The user may then choose to see plots of the mode shapes, which are automatically saved to the results folder for the user's convenience.

Following the modal analysis, the user is presented with frequency spectra presenting the amplitudes of the input excitations. The fundamental natural frequency is plotted on the same axes. This plot of the Fast Fourier Transform (FFT) of the excitation is of great significance and should be examined by the user. If the fundamental frequency coincides with a strong frequency component of the excitation, as shown in Figure 4.1, the structure will experience resonance, and the effects of the dynamic load will be more severe. In some instances, when the natural frequencies are spaced away from the peak excitation frequencies, the structure may exhibit milder response, and a hybrid simulation may not be worthwhile. In that case, the user may wish to terminate the analysis and adjust the input excitation waveforms or structural model to produce conditions more favorable to a strong dynamic response before restarting the main analysis routine.

The linear modal analysis results are also used to aid in establishing proportional

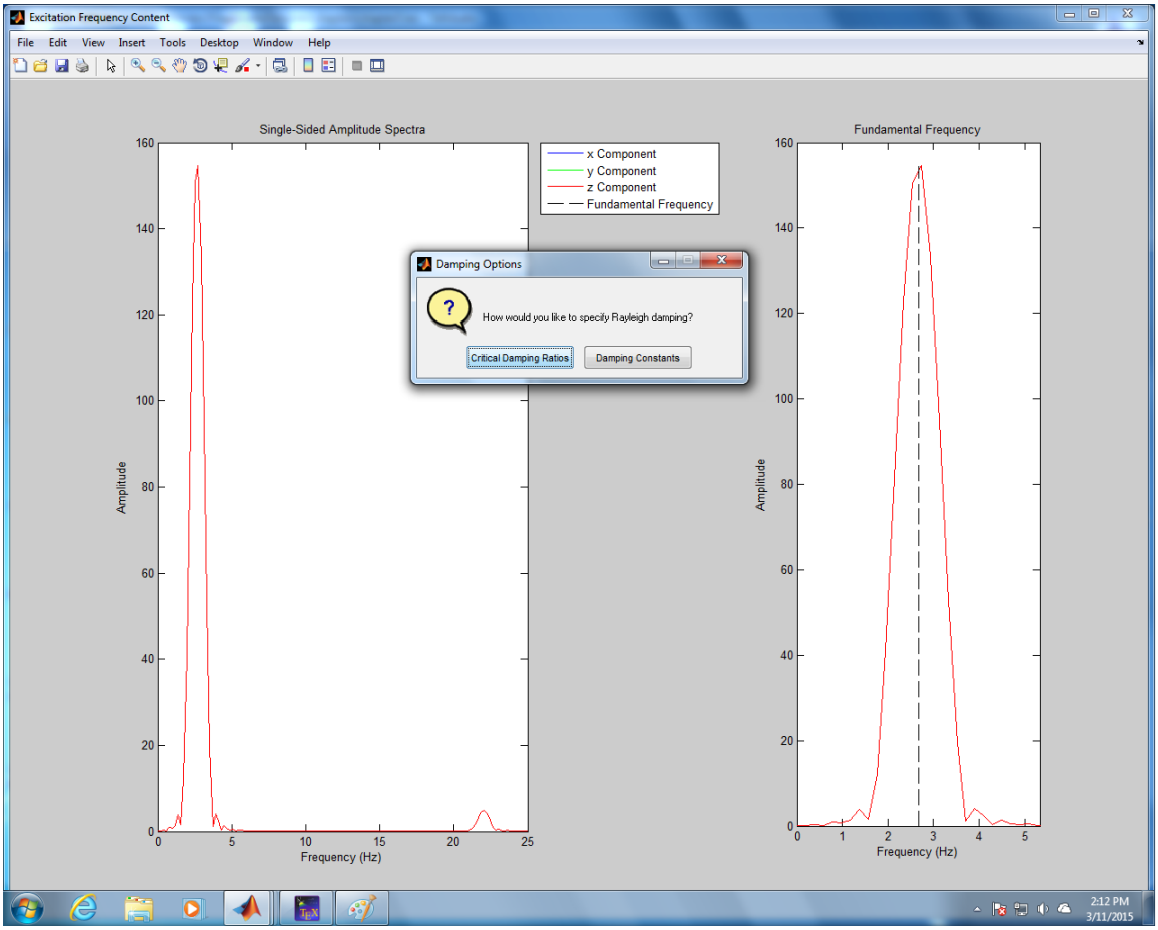



FIGURE 4.1: Excitation frequency plot, with an overlay of the first natural frequency

damping for the model via the function *DampingSetup.m*. The user may either specify the stiffness-proportional damping coefficient, $\alpha^c$, and mass-proportional damping coefficient, $\beta^c$, directly or request that the function calculate them using a pair of frequencies and corresponding desired damping ratios. Note that the $\alpha^c$ and $\beta^c$ used here are distinct from and unrelated to the $\alpha$ and $\beta$ appearing in the equations of Chapter 3. The former are used for modal damping, whereas the latter are used as parameters for the integration scheme. A generic figure is provided to the user to aid in the wise selection of the modal damping parameters, as shown in Figure 4.2. Following the selection of the damping parameters, another similar figure is generated

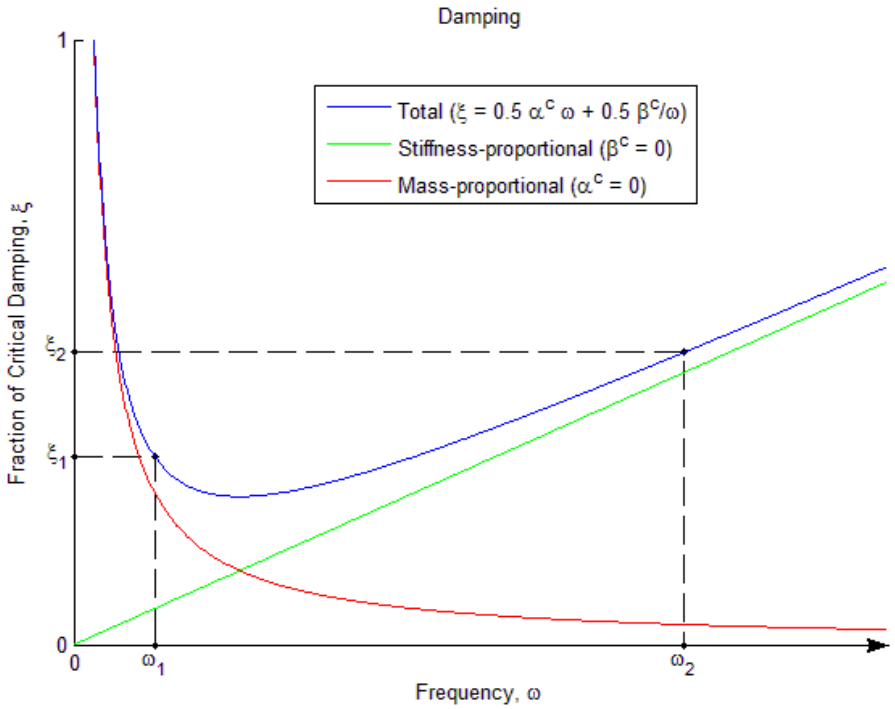



FIGURE 4.2: Proportional damping and critical damping ratios as functions of frequency, adapted from textbook [Cook, 1974]

to graphically represent the effect of the prescribed proportional damping over a range of frequencies. If the user is displeased with the damping model, he or she may respecify the coefficients immediately. Otherwise, the figure is saved in the results folder.

Once all inputs have been provided by the user, the function *IntegScheme.m* is called, and the simulation begins. Upon conclusion of the experiment, displacement time history plots are generated and stored in the results folder, which also contains the text file of nodal displacements, velocities, and accelerations recorded throughout the simulation.

## 4.2 Integration Scheme

The second and most fundamental tier of the HSF software is the integration scheme. In this section, the integration scheme of the HSF software is presented in algorithm form (Algorithm 2). The steps shown here closely follow the process encoded in the MATLAB code provided in Appendix E. This section is intended to provide an overview of the process, along with many of the equations used therein. In Section 4.3, the finer details and subtleties of the hybrid-specific portion of the process are presented.

**Algorithm 2** Integration Scheme

**Inputs:** stiffness-proportional damping ($\alpha^c$), mass-proportional damping ($\beta^c$), numerical damping parameter ($\alpha$), time vector ($\{t\}$), excitation forces ($Fexcite$), $jlimit$, $tol$, $\mathbf{M}$, $Specimen$, $Coordinates$, $\{E\}$, $\{A\}$

**Outputs:** .txtfile, $runinfo$

1: $n_{max} = length(\{t\}) - 1$
2: $\beta = (1-\alpha)^2/4$
3: $\gamma = (1-2\alpha)/2$
4: $\Delta t = t_1 - t_0$ (indirect result of $Fs$ in Algorithm 1)
5: .txtfile $\leftarrow t_0$, $\mathbf{d}_0$, $\mathbf{v}_0$, $\mathbf{a}_0$
6: **for** $n = 1$ to $n_{max}$ **do**
7: $\quad \Delta\mathbf{f} \leftarrow Fexcite$ (external force increment determined from excitations)
8: $\quad \Delta\mathbf{a} = \mathbf{0}$
9: $\quad \delta\mathbf{a} = \mathbf{0}$
10: $\quad$ **do** Iterations (Algorithm 3) to achieve equilibrium on nonlinear hybrid model
11: $\quad$ Retrieve contents of $Result$ from Iterations (Algorithm 3)
12: $\quad \mathbf{a}_n = \mathbf{a}_{n-1} + \Delta\mathbf{a}$ (acceleration vector)
13: $\quad \mathbf{v}_n = \mathbf{v}_{n-1} + \Delta\mathbf{v}$ (velocity vector)
14: $\quad \mathbf{d}_n = \mathbf{d}_{n-1} + \Delta\mathbf{d}$ (displacement vector)
15: $\quad \mathbf{r}_n = \mathbf{r}_{n-1} + \Delta\mathbf{r}$ (restoring force vector)
16: $\quad \mathbf{f}_n = \mathbf{f}_{n-1} + \Delta\mathbf{f}$ (external force vector)
17: $\quad \{f_{x,n}\} = \{f_{x,n-1}\} + \{\Delta f_x\}$ (array of member forces)
18: $\quad$ .txtfile $\leftarrow t_n$, $\mathbf{d}_n$, $\mathbf{v}_n$, $\mathbf{a}_n$
19: $\quad runinfo_n \leftarrow t_n$, $n$, $j$, $f^E_{x,n}$
20: **end for**

Within the integration scheme, initial estimates for the state vectors $\mathbf{a}$, $\mathbf{v}$, and $\mathbf{d}$ are made at the start of each time step. Using Algorithm 3, those estimates are refined iteratively to satisfy the equation of motion to within the user-specified tolerance. The resulting state vectors for the time step are then written to a text file, and the process is repeated.

Each iteration (Algorithm 3) begins with an adjustment of the incremental acceleration vector, $\Delta\mathbf{a}$, by an amount $\delta\mathbf{a}$, determined from the unbalanced force in the previous iteration as described in Section 3.3 with Equations 3.19, 3.20, and 3.21. For the first iteration, $\delta\mathbf{a}$ is taken as the zero vector. Following the adjustment of $\Delta\mathbf{a}$, the incremental displacement and velocity vectors are also adjusted. With the new incremental displacement vector, the geometry of the structure must be updated, along with all the variables that are affected by it, including member forces and the stiffness matrix. Since the damping matrix is constructed from the stiffness and mass matrices, the damping matrix is updated as well.

With the revised parameters, the incremental restoring force vector may be computed. In small-displacement linear analysis, the restoring force vector would be computed as the product of the stiffness matrix and the displacement vector. In the case of the HSF software routine, the incremental restoring force vector is computed using the updated member forces acting in the deformed configuration instead. In the *NodalEquivalent* subfunction of *IntegScheme.m*, each internal axial member force is transformed from local to global coordinates to generate an equivalent restoring force vector acting on the global degrees of freedom. The sum of the contributions from each of the members produces the equivalent restoring force. The incremental restoring force vector is then

**Algorithm 3** Iterations

**Inputs:** everything in Algorithm 2
**Outputs:** $\Delta\mathbf{a}$, $\Delta\mathbf{v}$, $\Delta\mathbf{d}$, $\Delta\mathbf{r}$, $\{\Delta f_x\}$, $k_n^E$, $\{L_n\}$

1: **for** $j = 1$ to $jlimit$ **do**
2: $\Delta\mathbf{a} = \Delta\mathbf{a} + \delta\mathbf{a}$ (Predictor acceleration, $\delta\mathbf{a} = \{0\}$ for $j = 1$)
3: $\Delta\mathbf{v} = \Delta t\mathbf{a}_{n-1} + \gamma\Delta t\Delta\mathbf{a}$ (Predictor velocity)
4: $\Delta\mathbf{d} = \Delta t\mathbf{v}_{n-1} + \frac{1}{2}\Delta t^2\mathbf{a}_{n-1} + \beta\Delta t^2\Delta\mathbf{a}$ (Predictor displacement)
5: $Coordinates = Coordinates + \mathbf{d}_{n-1} + \Delta\mathbf{d}$ (Update coordinates of nodes)
6: $\{L_n\}, \{\mathbf{T}_n\} \leftarrow Coordinates$ (Update lengths and transformation matrices)
7: **for all** members except *Specimen* **do**
8: $f_{x,n} = \frac{EA+f_{x,n-1}}{L_{n-1}}(L_n - L_{n-1}) + f_{x,n-1}$ (calculate internal member force)
9: **end for**
10: **for** *Specimen* **do**
11: $f_{x,n}$, $k_n^E \leftarrow$ *ExpSub.m* (Algorithm 4)
12: **end for**
13: $\{\Delta f_x\} = \{\Delta f_{x,n}\} - \{f_{x,n-1}\}$
14: $\mathbf{K} \leftarrow KTotal$ (Update $\mathbf{K}$ for geometric nonlinearities)
15: $\mathbf{C} = \alpha^c\mathbf{K} + \beta^c\mathbf{M}$ (Update $\mathbf{C}$ for only geometric nonlinearities)
16: $\mathbf{K} \leftarrow KTotal$ (Update $\mathbf{K}$ for all nonlinearities)
17: $\mathbf{r}_n \leftarrow NodalEquivalent(\{f_{x,n}\},\{\mathbf{T}_n\})$
18: $\Delta\mathbf{r} = \mathbf{r}_n - \mathbf{r}_{n-1}$
19: **if** $j = 2$ **then**
20: $Result \leftarrow \Delta\mathbf{a}$, $\Delta\mathbf{v}$, $\Delta\mathbf{d}$, $\Delta\mathbf{r}$, $\{\Delta f_x\}$, $k_n^E$, $\{L_n\}$
21: **end if**
22: $\hat{\mathbf{M}} = \mathbf{M} + (1+\alpha)(\gamma\Delta t\mathbf{C} + \beta\Delta t^2\mathbf{K})$
23: $\hat{\mathbf{f}} = \mathbf{f}_{n-1} - \mathbf{M}(\mathbf{a}_{n-1} + \Delta\mathbf{a}) - \mathbf{C}\mathbf{v}_{n-1} - \mathbf{r}_{n-1} + (1+\alpha)(\Delta\mathbf{f} - \mathbf{C}\Delta\mathbf{v} - \Delta\mathbf{r})$
24: $\delta\mathbf{a} = \hat{\mathbf{M}}^{-1}\hat{\mathbf{f}}$
25: **if** $j > 1$ **then**
26: $norm = \sqrt{\frac{1}{length(\mathbf{d})}\sum\left(\frac{\beta\Delta t^2\delta\mathbf{a}}{\max(\|\Delta\mathbf{d}\|)}\right)^2}$
27: **if** $tol > norm$ **then**
28: $Result \leftarrow \Delta\mathbf{a}$, $\Delta\mathbf{v}$, $\Delta\mathbf{d}$, $\Delta\mathbf{r}$, $\{\Delta f_x\}$, $k_n^E$, $\{L_n\}$
29: **go to** 32
30: **end if**
31: **end if**
32: **end for**

computed as the difference between the new restoring force and the restoring force from the beginning of the time step.

Once the incremental restoring force vector has been computed, all the quantities are available for determining the unbalanced force and necessary adjustment, $\delta\mathbf{a}$, for the incremental acceleration vector. The resulting vector is used to check convergence criteria based on a modified Euclidean norm of displacements as expressed in Equation 12.23 of *Matrix Structural Analysis* [McGuire et al., 2000].

For cases where the user-specified iteration limit is reached and the convergence criteria are still not satisfied, the $\alpha$-OS predictor-corrector method is used. This task requires no additional computation, since the first iteration of each time step is simply a predictor-corrector step. The results from the second iteration of each time step are temporarily held in memory as fields of the structure `Result`. If at any iteration the convergence check is satisfied, `Result` is overwritten and the "for" loop exited.

## 4.3 Experimental Substructure

The third tier of the HSF is embedded within the iteration loop. As indicated in Algorithm 3, the experimental restoring force and the analytical internal member forces are obtained separately. The experimental restoring force is obtained from direct measurement, while the member forces in the analytical substructure are produced using Equation 3.2 in the iteration loop (Line 8 of Algorithm 3). For the experimental substructure, the function *ExpSub.m* (Appendix F) is used to either produce a new measurement or an estimate for an iteration via polynomials fitted to the most recent data points. The operations of *ExpSub.m* (Algorithm 4) are the focus of this section.

**Algorithm 4** Experimental Substructure

**Inputs:** $L_n - L_0$, $\Delta t$, $\{t\}$, $n$, $j$, $tol$
**Outputs:** $f^E_{x,n}$, $k^E_n$

**if** $j = 1$ **then**
  Impose $\Delta d_{desired}$ physically on experimental substructure
  Get $\Delta d_{achieved}$ and $\Delta f_{measured}$ from direct measurement
  **if** $n = 1$ **then**
    Get linear fit coefficients for actuator $d$ and $f$ versus $t$
  **else**
    Get quadratic fit coefficients for actuator $d$ and $f$ versus $t$ for latest points
  **end if**
  **if** $\Delta d_{achieved} > threshold$ **then**
    $k^E_n = \Delta f_{measured} / \Delta d_{achieved}$ (Update stiffness estimate)
  **else**
    $k^E_n \leftarrow k^E_{n-1}$ (Revert to trusted stiffness estimate)
  **end if**
  $f_{x,n} = f_{measured} + k^E_n(\Delta d_{desired} - \Delta d_{achieved})$ (Update restoring force estimate)
**else**
  **if** $n > 1$ **and** $d_{fit}(t)$ is approximately linear **then**
    Neglect 2nd-order term in $d_{fit}$
  **end if**
  Solve $d_{fit}(t) = d_{desired}$ for $t$, including complex solutions
  Find the magnitude of the difference between $t_n$ and each $t_{sol}$ by

$$t_{diff} = \sqrt{\|t_{sol,real} - t_n\|^2 + \|t_{sol,imag} i\|^2}$$

  Discard all $t_{sol}$ except the one with the smallest $t_{diff}$
  **if** $t_{diff} < 2\Delta t$ **then**
    $f_{x,n} = \|f_{fit}(t_{sol})\| \times \text{sign}(R_e\left(f_{fit}(t_{sol})\right))$
  **else**
    **if** $n = 1$ **then**
      $f_{x,n} = k^E_0 d_{desired}$
    **else**
      no solution
      $f_{x,n} = f_{measured} + k^E_n(\Delta d_{desired} - \Delta d_{achieved})$
    **end if**
  **end if**
  **if** $\Delta d_{desired} > threshold$ **and** solution was found **then**
    $k^E_n = \Delta f_{estimated} / \Delta d_{estimated}$ (Update stiffness estimate)
  **else**
    $k^E_n \leftarrow k^E_{n-1}$ (Revert to trusted stiffness estimate)
  **end if**
**end if**

### 4.3.1 Description

*ExpSub.m* is used to determine the restoring force and equivalent stiffness of the experimental substructure during the hybrid simulation. This function is called on each iteration of each increment of a hybrid simulation. On the initial iteration of an increment, the procedure is routed into either the *Exp* or *ExpSim* subfunction of *ExpSub.m*, depending on whether a numerical verification or an actual experiment is being performed. The corresponding subfunctions produce either actual or synthetic measurement outputs that are stored in the global variables `dhist` and `rhist` in terms of the base units of the particular simulation ($N$ and $m$ or $lb$ and $ft$), with the axial displacements scaled for the length of the entire experimental member if the physical specimen is only a segment of the whole member. The first time these functions are called in an experiment, they also record the initial readings for displacement and restoring force of the experimental substructure. In the subfunction *Exp* of *ExpSub.m*, the desired displacement is passed to the hardware, and experimental measurements of restoring force are acquired by reading the signal from the load cell. This process is demonstrated through experimental application in Chapter 6. In the subfunction *ExpSim* of *ExpSub.m*, theoretical behavior of the experimental substructure is modeled and used to produce synthetic restoring forces due to prescribed displacements. This function could also be modified to simulate systematic or random error in measurement, if desired.

In hybrid simulations with nonlinear behavior, the chronology of loading is important to the accuracy of the experiment, since loading can produce irreversible damage

in the specimen. Consequently, while iterations can be performed on the analytical substructure without loss of accuracy, iterations on the experimental substructure are not feasible. This challenge has been one of the major obstacles to implementing hybrid simulation of highly nonlinear structures in recent years. Mosqueda and Ahmadizadeh addressed this limitation by proposing a numerical technique for carrying out virtual iterations on the experimental substructure [Mosqueda and Ahmadizadeh, 2011]. On non-initial iterations of an increment, second-order polynomials (fitted to the latest experimental data points) are used to interpolate or extrapolate the experimental restoring force corresponding to the desired displacement of the experimental substructure. It is important to note that the forces and displacements are not directly related to one another in this method. Instead, they are related to one another through their relationships to time, as shown in Figure 4.3. One of the important advantages of this

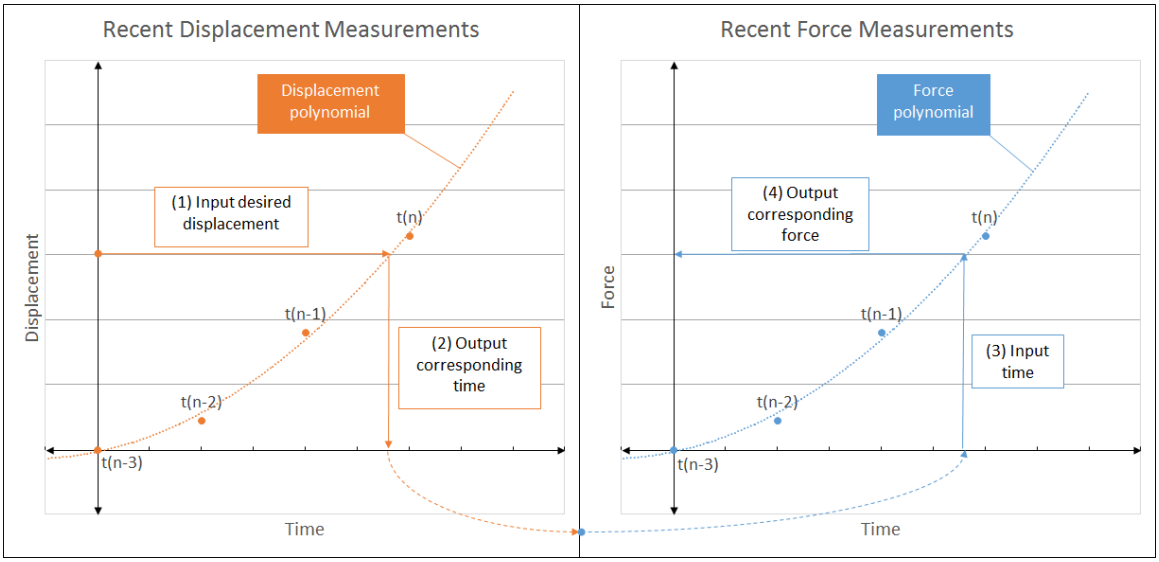



FIGURE 4.3: Estimation procedure using fitted polynomials, adapted from Mosqueda and Ahmadizadeh [Mosqueda and Ahmadizadeh, 2011]

approach is that "the effects of specimen nonlinearities will be less pronounced on time histories compared to force-displacement curves, providing better quality curve fitting," [Ahmadizadeh, 2007]. This approach also has several benefits in the context of real-time simulation, but real-time simulation is beyond the scope of this thesis.

First, the polynomial fitted to the displacement-time data is used to estimate the time at which the desired iterative displacement would theoretically be achieved in the experimental substructure. Then, the polynomial fitted to the force-time data is used to estimate the force corresponding to that time.

*ExpSub.m* is not only responsible for obtaining and iterating on the restoring force from the experimental substructure, but also for updating the estimated stiffness coefficient, $k_n^E$, for the experimental substructure. The stiffness for the experimental substructure is approximated as the secant stiffness exhibited over the time step, computed by the change in the restoring force divided by the change in length of the member, since the experimental substructure for this application is a single axial element. As pointed out by Ahmadizadeh, updating the stiffness in this way could be inaccurate if the applied incremental displacement is too small, since measurement noise would have a relatively large magnitude compared to the force and displacement that are intended [Ahmadizadeh, 2007]. For this reason, a criterion is adopted that the incremental displacement must be greater than a specified threshold in order for $k_n^E$ to be updated in any given iteration. If the threshold is not exceeded in an iteration, $k_n^E$ reverts to its value from the end of the previous time step $k_{n-1}^E$ until an iteration in which the threshold is exceeded. Guidance on the recommended threshold is provided by Ahmadizadeh [Ahmadizadeh, 2007]. This threshold is established in the first call

of *ExpSub.m*, using the outputs of the subfunction *hardwaresetup* or *hardwaresetupsim* of *ExpSub.m*, depending on whether a numerical verification or an actual physical experiment is being performed.

### 4.3.2 Safeguards in Implementation of Estimations Using Polynomials

To estimate forces using fitted polynomials while limiting the opportunities for erroneous estimates, several safeguards have been implemented, mostly following the recommendations of the original proposers of this technique. These safeguards are described in the following subsections, and their implementations can be identified in the "if" statements of Algorithm 4.

#### 4.3.2.1 Insufficient Data Points

At the start of an experiment, there will be too few data points to generate quadratic fits of the force and displacement responses. In this case, linear fits are used instead. Another obstacle that often arises towards the start of an experiment is that the command displacement may not change for the first couple of steps, due to the commonly-used initial conditions of zero acceleration and velocity. Under these conditions, acceleration will become non-zero in the first time step, and velocity will become non-zero in the second time step. In the third time step, displacement will finally change. For cases such as these, iteration on the experimental displacement cannot be achieved by fitting polynomials to the data, since the displacement-time data exhibits a constant trend. Instead, the theoretical initial stiffness of the experimental substructure is used to produce the restoring force in the member from an assumed linear elastic behavior, which is valid since displacement should be small. Alternatively,

the initial stiffness estimate could be improved based on a direct measurement of the initial linear elastic stiffness of the experimental substructure prior to the experiment.

#### 4.3.2.2 Linear Displacement-Time Trend

In some situations, recently commanded displacements may coincidentally (or perhaps by design) exhibit a linear or nearly-linear trend. In this case, a second-order polynomial will either be inaccurate or poorly scaled. For this reason, the polynomial solution procedure needs to be able to default to solving a linear fit in situations where a quadratic fit is less appropriate. To detect such situations, the first coefficient of the quadratic fit is divided by the second coefficient. If this value is small, say less than $\frac{1}{10000}$, it can be safely assumed that a linear fit would be a more appropriate option. At that point, the second and third coefficients of the quadratic polynomial can be used as the first and second coefficients of a linear polynomial. This issue does not affect the process of solving the force-time polynomial, since that polynomial is solved only in the forward sense by substituting a given time value into the fitted quadratic equation. For that use of the second-order polynomial model, scaling of coefficients does not present computational challenges.

#### 4.3.2.3 Excessive Extrapolation

As recommended by the original proposers of this method, the window of time for acceptable solutions should be restricted to within two time steps of the time at the end of the current time step in order to prevent excessive extrapolation of experimental data [Mosqueda and Ahmadizadeh, 2011]. If no solution exists within the acceptable time window, the most recent stiffness estimate, $k_n^E$, of the experimental substructure

is used to make the adjustment from the measured restoring force (corresponding to the applied displacement) to the estimated restoring force (corresponding to the desired displacement) to achieve equilibrium during iterations. Implementation of this safeguard can be found at the end of the function *ExpSub.m* presented in Appendix F.

#### 4.3.2.4 Complex Solutions

The last complication that may arise in using quadratic fits to estimate displacements and forces involves complex roots. In some cases, there may not be a real value of time for the desired displacement on the fitted polynomial, namely at points of displacement reversal [Ahmadizadeh, 2007]. This raises the question of what, if anything, to do with complex solutions. Ahmadizadeh's recommendation is to accept the complex solution of the displacement polynomial, if the imaginary component is small, and to compute the value of the force polynomial for that complex-valued time [Ahmadizadeh, 2007]. This is actually an extension of the restriction on the time window discussed previously. In the case of complex roots, excessive extrapolation can be avoided by limiting the accepted solution times, $t_{sol}$, to those falling within a time "radius," $r$, of length $2\Delta t$ of the current time, $t_n$, as shown in Equation 4.1. Decomposing $t_{sol}$ into real and imaginary components ($t_n$ only has a real component), this radius can be determined as

$$r = \sqrt{\|t_{sol,real} - t_n\|^2 + \|t_{sol,imag}i\|^2} < 2\Delta t \tag{4.1}$$

By limiting the magnitude of the difference between the current time and the solution time, the spirit of the time window restriction is fulfilled in the context of complex numbers in a non-arbitrary way. Once the force polynomial has been evaluated

for the complex-valued time, the accepted force estimate is taken as the absolute value of the complex-valued force, multiplied by the sign of the real component of the complex-valued force. While real roots are preferable, this procedure allows the integration scheme to proceed without the dissipation of energy associated with the additional displacement.

# CHAPTER 5: VERIFICATION OF FRAMEWORK

Prior to performing experimental validation, verification of the HSF was required to ensure that the scheme was properly implemented and functioning as intended. For this study, the verification of the HSF was performed by comparison with SAP2000 analysis over a series of both linear and nonlinear cases. First, the options within the SAP2000 nonlinear direct time-history integration were set in such a way as to carefully imitate the details of the integration scheme used in the HSF. After ensuring this consistency, the basic integration scheme used in the HSF was verified against the SAP2000 software results for an analysis of a simple two-element structure under an applied dynamic load. Then, the routine for the HSF including experimental substructuring was verified using the same structural model to ensure that the hybrid-specific portion of the HSF performed appropriately as a substitute for the basic procedures associated with the analytical substructure. Finally, verification of the HSF software routine was performed on a large three-dimensional model subject to base excitation to ensure accurate analysis after expansion of model complexity.

## 5.1 Verification by Comparison Against Commercial Software

In order to verify the proper performance of the MATLAB-based HSF, a reliable baseline simulation from a trusted and verified code must be established for comparison. Each aspect of the analysis procedure is considered, from material properties and

geometry assignments to material nonlinearity models and aspects of the integration scheme, iterations, and convergence criteria. Nearly every aspect of the analysis performed in SAP2000 was mimicked using the MATLAB-based HSF, in accordance with *CSi Analysis Reference Manual* [Computers and Structures, 2009]. Once the results from the two analysis packages were in close agreement, it was concluded that the HSF correctly accounts for the various linear and nonlinear aspects of the structural analysis.

### 5.1.1 Usage of Application Programming Interface

The most reliable method of verification using SAP2000 is to operate SAP2000 via the CSi Open Application Programming Interface (OAPI). The OAPI is an effective tool to eliminate user error or forgetfulness, since a program may be written to carry out all of the modeling and analysis tasks according to a few inputs rather than having to manually set up the SAP2000 model and analysis through the graphical user interface. Such a program was developed and used in all verifications for the HSF presented in this thesis. The program is provided in Appendix G.

### 5.1.2 Units, Material Properties, and Geometry Assignments

As in any structural analysis software, values of inputs provided to SAP2000 must reflect units consistent with those used by the HSF throughout the modeling and analysis process. Material properties (Young's modulus, $E$, and mass density, $\rho$) and section properties (cross-sectional area, $A$) must be assigned for each element, as well as the geometry of the global model, end releases of the elements, and restraint assignments for boundary conditions of the model.

### 5.1.3 Link Element for Specimen

A link element should be used in place of a frame element to represent the experimental member (or experimental substructure) during verification of cases with nonlinear material models. To model a truss member with a link, only the axial degree of freedom of the link should be activated, and none of the degrees of freedom of the link element are fixed. Nonlinearity should be enabled for the axial degree of freedom, and the effective stiffness for linear analyses should be set as $EA/L$ to ensure accurate modal analysis for initial model comparison. Likewise, link weight and mass should be specified to ensure that the link element contributes properly to the mass matrix, although the weight may be unused if self-weight is neglected in the analysis. In order to establish the nonlinear material behavior of the experimental member, a multilinear material model can be used to provide an idealized nonlinear elastic hysteresis model. The simplest multilinear material model can be constructed as a bilinear material model using five ordered pairs, as shown in Figure 5.1. The slope of the line connecting

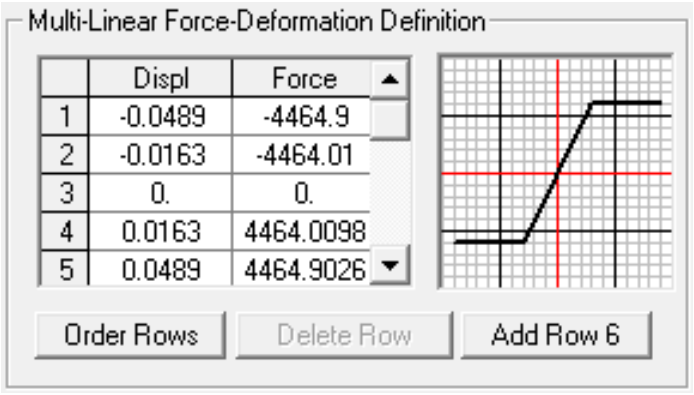



FIGURE 5.1: Multilinear elastic material model assignment in SAP2000

the third and fourth ordered pairs will reflect the initial stiffness of the experimental member. The post-yield incremental stiffness will be defined by the slope of the line connecting the fourth and fifth ordered pairs. The first and second ordered pairs are normally assigned to be antisymmetric to the fourth and fifth ordered pairs. SAP2000 will not accept a zero-stiffness segment, so in cases where zero stiffness is specified, 0.01 percent of the initial stiffness should be used instead in both the MATLAB and SAP2000 assignments to ensure model consistency.

### 5.1.4 Damping Matrix Updates

For a given time step, the stiffness matrix used in SAP2000 for nonlinear direct time history load cases is constructed using the contributions from any linear or nonlinear material model, as well as contributions from geometric nonlinearities for the time step, if large-displacement analysis is enabled. The proportional damping matrix, however, is computed on the basis of a stiffness matrix formed by the initial linear elastic stiffnesses and updated geometries of the elements, neglecting the material nonlinearity of the link element. This is documented in the *CSi Analysis Reference Manual* [Computers and Structures, 2009]. Since only this approach is available in SAP2000 for the construction of the damping matrix in nonlinear analysis, the same approach is used in the integration scheme implemented in the HSF. As a result, the integration scheme code contains a line for construction of the stiffness matrix with only nonlinear geometric effects, followed immediately by a line for the computation of the updated damping matrix. This stiffness matrix is used only for the formation of the proportional damping matrix. Subsequently, the stiffness matrix is reconstructed

with consideration of nonlinear material behavior in the experimental substructure, in addition to the geometric nonlinearity in the system.

### 5.1.5 Load Case Definitions

To ensure that consistent excitation time histories will be used in both SAP2000 and the developed code, time history functions are defined from text files containing joint forces or ground accelerations applied in each direction. Following the definition of these excitation functions, load cases are defined in SAP2000. In order to effectively mimic the procedure used in the HSF, SAP2000 load case definitions should include exactly the same proportional damping constants $\alpha^c$ and $\beta^c$, computed as discussed in Section 4.1, the same time step size $\Delta t$, the correct number of output time steps, the desired geometric nonlinearity setting (typically including large-displacement), the loads from the aforementioned functions as accelerations or loads in each direction, a consistent time integration method and accompanying integration scheme parameters (HHT with $\alpha = 0$ for this verification), and solution control parameters. The solution control parameters should include maximum and minimum substep sizes equal to the time step size (thereby disabling the default substepping feature in SAP2000), maximum constant-stiffness iterations of zero, maximum Newton-Raphson iterations equal to the iteration limit, relative iteration convergence tolerance equal to the desired tolerance, no event-to-event stepping, and the maximum number of line searches set to zero. This will produce an iteration procedure that matches that of the HSF nearly exactly. Unfortunately, the convergence criterion used in SAP2000 is force-based, whereas the criterion used in the HSF is displacement-based. However, this has not

been observed to have a noticeable effect on the converged solution.

## 5.2 Verification Cases for Material and Geometric Nonlinearities

To be certain that nonlinear analysis capabilities of the HSF software are adequately tested in the verification process, four cases of analysis of the same simple structure subject to the same loading are compared using direct time-history analysis in SAP2000. For simplicity of the results presented, the displacement response of a single joint is considered. Since the HSF is based on an iterative method, the load case used in SAP2000 must include iterations in order to be comparable. For this reason, all SAP2000 direct time-history verification analyses are run as nonlinear analyses. Due to the inability to toggle between linear and nonlinear material behavior within a nonlinear direct time-history load case, a work around is required for cases where the desired experimental member material behavior is linear elastic. To execute this informal linear analysis in SAP2000 while still using the nonlinear direct time-history load case, geometric nonlinearity is turned off, and the experimental member is modeled for linear elastic material behavior using a frame element rather than a link element. This analysis represents the linear elastic material, first-order (linear) geometric analysis (LMLG) case. A similar process is used to perform the analysis for the case in which linear elastic material behavior is considered with a second-order (nonlinear) geometric analysis (LMNLG). The other two analysis cases included in this verification are nonlinear material behavior with first-order geometric analysis (NLMLG) and nonlinear material behavior with nonlinear geometric analysis (NLMNLG). To ensure that nonlinear geometric effects and nonlinear material effects are significant for the

structure under the prescribed loading, the SAP2000 NLMNLG results should be confirmed to be distinct from the results of the other three cases. Additionally, the results of at least one of the two partially-nonlinear cases (NLMLG and LMNLG) should be distinct from the results of the purely-linear case (LMLG). For NLMLG and NLMNLG cases, the force or strain in the experimental member during the course of the analysis should also be confirmed to exceed the force or strain required to yield the member, preferably on numerous occasions throughout the analysis and by a significant margin. If these three requirements are not met, verification will be inconclusive at best and misleading at worst.

## 5.3 Verification of the Software Routine on a Hybrid Structure

At the most fundamental level, the HSF software routine is a software routine for numerical simulation, fitted with a special function to overwrite the analytical response of a substructure with the experimentally-observed response of that substructure. Thus, the majority of the HSF software routine can be verified by short-circuiting the polynomial-based pseudo-iteration technique and directly iterating on a virtual experimental substructure instead. An experiment simulator function is used within the HSF code to calculate the exact response of a virtual experimental substructure to prescribed displacements according to the nonlinear elastic material model. This is achieved with a multilinear curve similar to the one defining the multilinear link behavior in SAP2000 (see Figure 5.1). If any discrepancies exist between the results generated by SAP2000 and the HSF, they are used at this stage for debugging the integration scheme and supporting processes. This approach was instrumental for

identifying important subtleties of SAP2000's nonlinear direct time-history integration methods (such as updating the damping matrix at each time step while neglecting material nonlinearity in the damping matrix) without the added complications of implementing the polynomial iteration scheme.

Following the direct iteration step of verification, the polynomial-based pseudo-iteration technique was enabled for the verifications presented in the following subsections. Whereas the direct iteration procedure exactly produces the proper force for each iterative displacement of the experimental substructure, the polynomial estimation procedure is an approximation. Since SAP2000 performs iterations directly without consideration of experimental versus analytical substructures, SAP2000 lacks a feature analogous to the polynomial-based pseudo-iteration used in the HSF. Thus, exact agreement between the HSF and SAP2000 should not be expected when the polynomial estimations are used, but the results should still be close if the method is properly implemented. Successful verification of the HSF software routine demonstrates an acceptable level of accuracy and seamless execution of the estimation procedure.

### 5.3.1 Verification on a Simple Two-Dimensional Analytical Model

A useful consideration when selecting a structure for verification is the simplicity of the structure. As previously stated, the structure shown in Figure 5.2 was used for the verification of the HSF presented in this thesis. The horizontal member was selected as the experimental member. This structure, based on the problem described in Section 3.1, was selected primarily for its limited number of degrees of freedom and the dramatic difference in the cross-sectional areas of its two members. Such uneven

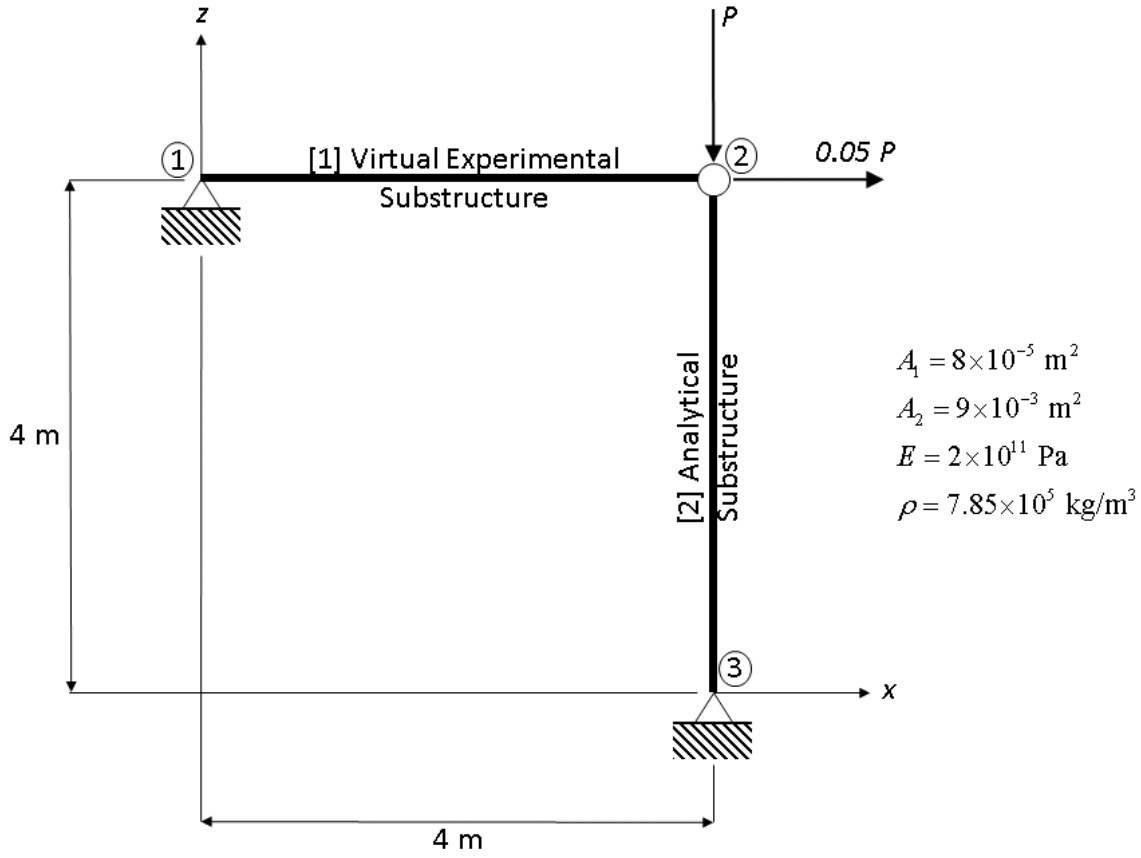



FIGURE 5.2: Simple two-dimensional truss model

distribution of stiffness lends itself to producing geometric and material nonlinearities and a clear difference in displacement amplitude for the two degrees of freedom, which can be informative in assessing the relative displacement error.

This structure has a fundamental natural frequency of 2.67 Hz, which occurs in a mode of vibration of the structure in the horizontal direction. The second natural frequency is 28.27 Hz, corresponding to the vertical mode of vibration of the structure. The structure was loaded with a windowed sinusoidal signal, $P$, applied in both directions with a fixed proportionality scale as shown in Figure 5.2. Figure 5.3 is a

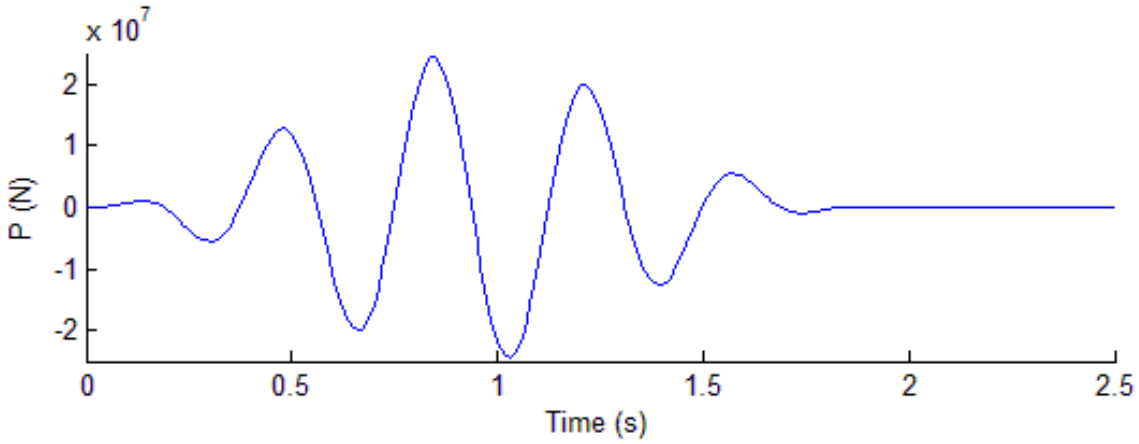



FIGURE 5.3: Force-time plot for load $P$ on two-element structure for verification

plot of $P$ as a function of time for this dynamic load. The frequency of the sinusoid was chosen to correspond to the fundamental natural frequency of the structure.

The stiffness-proportional and mass-proportional damping coefficients were set at 0.06 and 0.1, respectively, resulting in relative damping factors of 0.099 for the fundamental mode and 0.850 for the second mode. Relative displacement convergence tolerance was set at $1 \times 10^{-6}$ with a maximum of fifty iterations per time step, and the numerical dissipation parameter, $\alpha$, was set to zero. The material model for the experimental member was defined as bilinear with a yield stress of 2.25 GPa, an initial modulus of elasticity of 200 GPa, and a post-yield incremental stiffness of zero. The input file for this truss is provided in Appendix B.

After running analyses for LMLG, LMNLG, NLMLG, and NLMNLG cases, the SAP2000 results for the various cases were compared against one another to confirm that geometric and material nonlinearities are significant for the specified loading. Figures confirming this are provided in Appendix H. The stress-strain curves for

each of the four cases are provided in Appendix H as well, confirming that the member does indeed yield under the given loading when material nonlinearity is considered. The plots provided in Figures 5.4 and 5.5 show the agreement between the displacement time histories generated using the HSF and those from corresponding SAP2000 analyses. Although there is some discrepancy between the two software programs for the cases with nonlinear material behavior, the source of the difference seems to be the approximation of the stiffness in the HSF software routine for the time steps in which the experimental member transitions between the yielded and the linear elastic segments of the stress-strain curve, since SAP2000 need not make such approximations. This topic is further discussed in Section 5.3.3.

In addition to confirming the accuracy of the results, the usage of iterations in the analysis was checked. As indicated by Figure 5.6, which presents iteration counts for the nonlinear material, nonlinear geometry case, the time steps typically did not converge within the minimum two iterations except after the sinusoidal excitation subsided. During the excitation, most time steps required four iterations to converge to within the desired tolerance. This demonstrates both the need for and the effectiveness of the iterative integration scheme. In linear analyses, iterations are unnecessary. Regular convergence within two iterations (one as a predictor, one as a corrector) would erroneously suggest that iterations are unnecessary for nonlinear analyses as well. The fact that all time steps in this verification converged in fewer than the maximum number of iterations indicates that the iterative method effectively fulfills its purpose for this simple verification model.

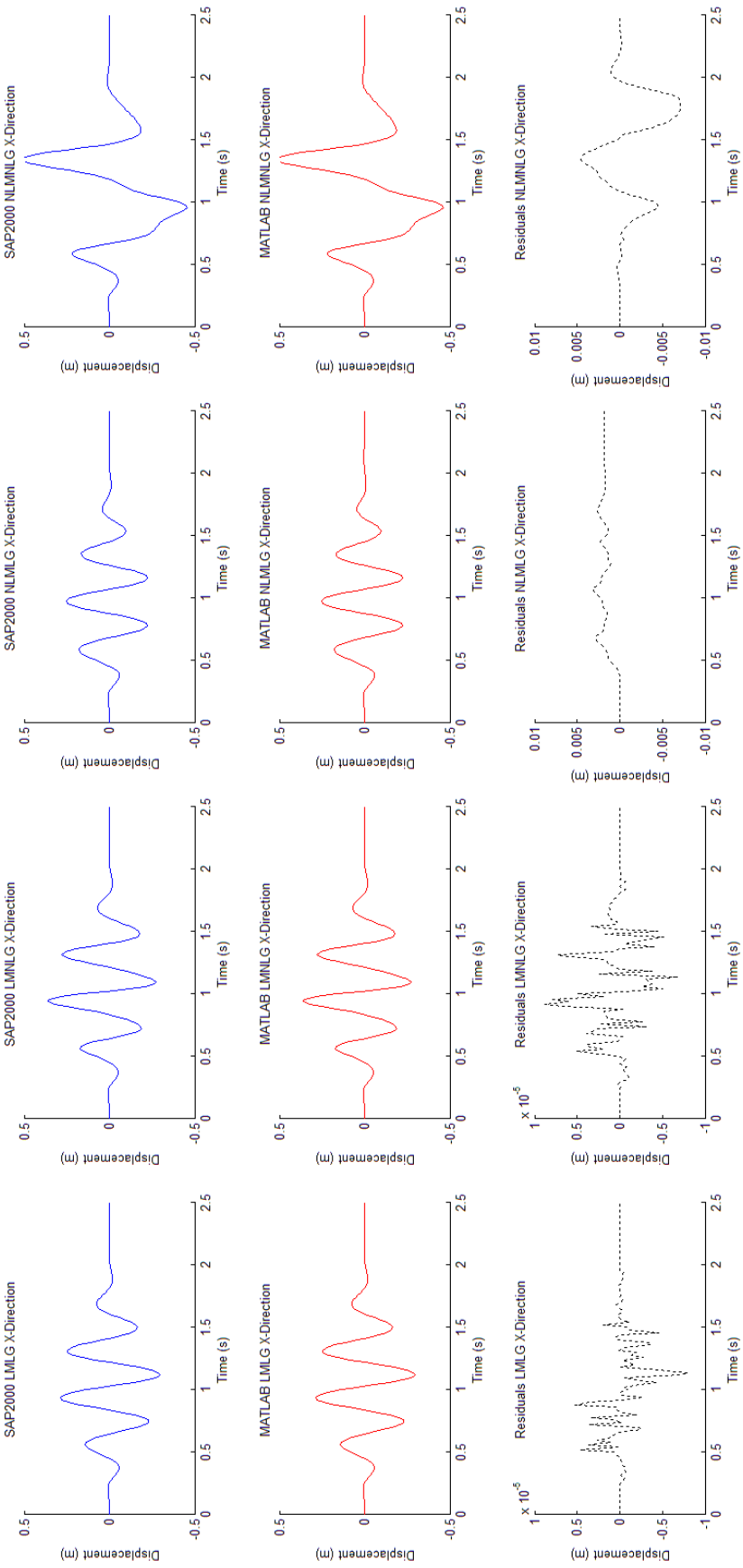



FIGURE 5.4: Displacement time histories for verification cases of planar model ($x$ direction)

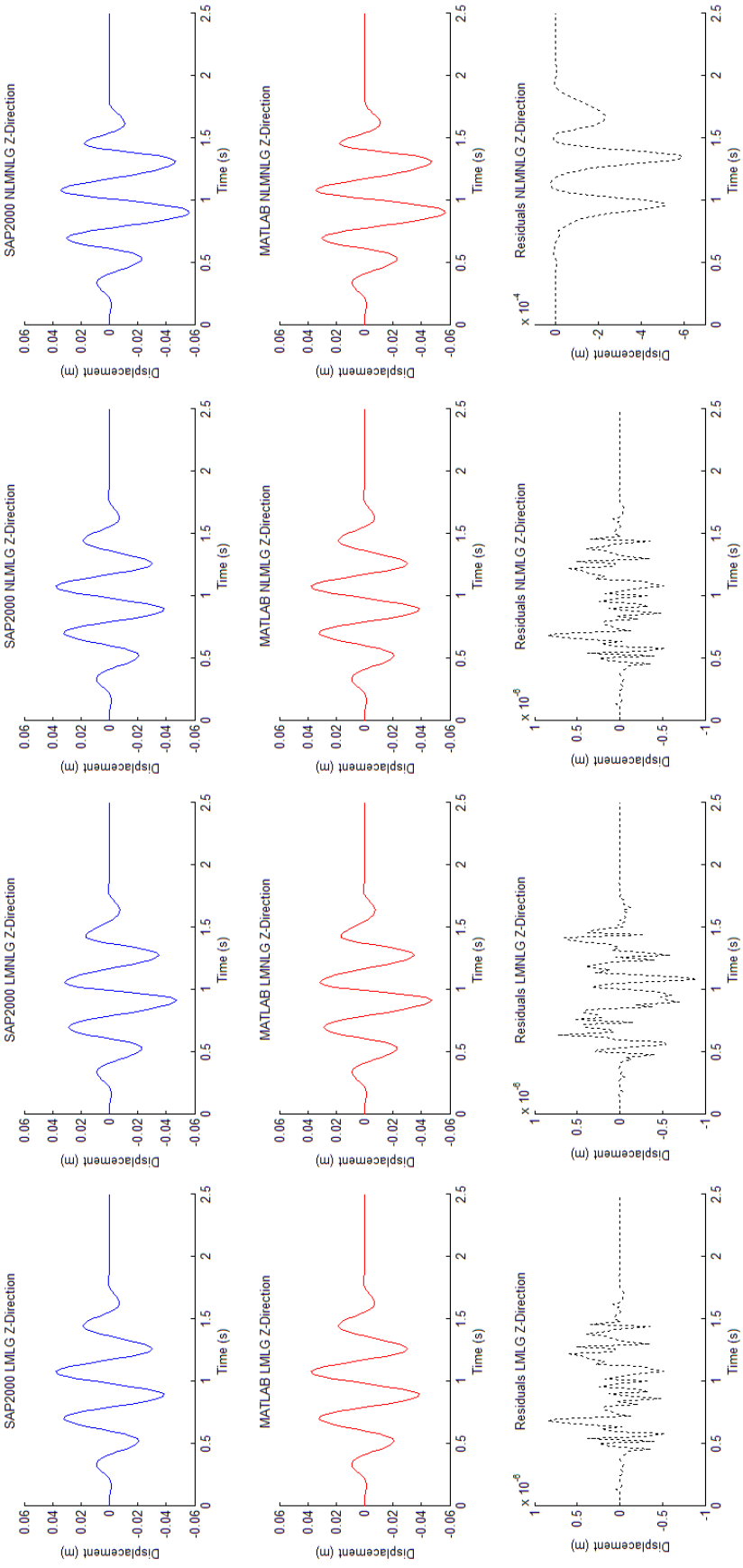



FIGURE 5.5: Displacement time histories for verification cases of planar model ($z$ direction)

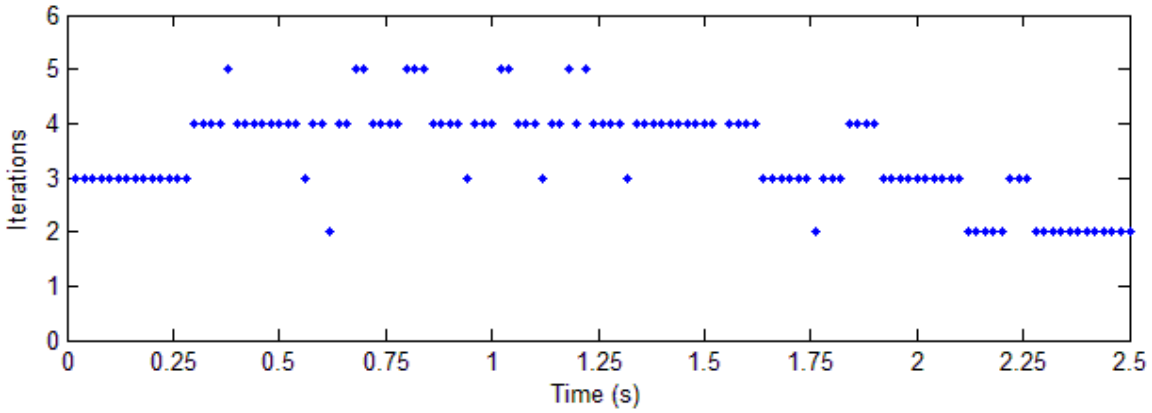



FIGURE 5.6: Iterations required for convergence throughout time history of NLMNLG case

### 5.3.2 Verification on a Three-Dimensional Analytical Model with Ground Motion Excitation

The verification presented in the previous subsection confirms that the integration scheme performs accurately for a planar structure of limited degrees of freedom. Although the American Society of Civil Engineers (ASCE) has stated that "transmission structures need not be designed for ground induced vibrations caused by earthquake motion," [Wong and Miller, 2009, p. 69] seismic design and analysis are the most common applications of hybrid simulation in contemporary research efforts. To explicitly demonstrate the applicability of the HSF to this important field of study and provide a verification case on a space truss, the structure shown in Figure 5.7 was subjected to ground accelerations from a segment of time history data for the 1994 Northridge earthquake [PEER, 2013] as an additional verification. The input file for this three-dimensional tower is given in Appendix I. Plots of the applied ground accelerations and the corresponding amplitude spectra for the $x$, $y$, and $z$ directions are presented in Figure 5.8. The magnitude of the excitation was scaled by 10 to

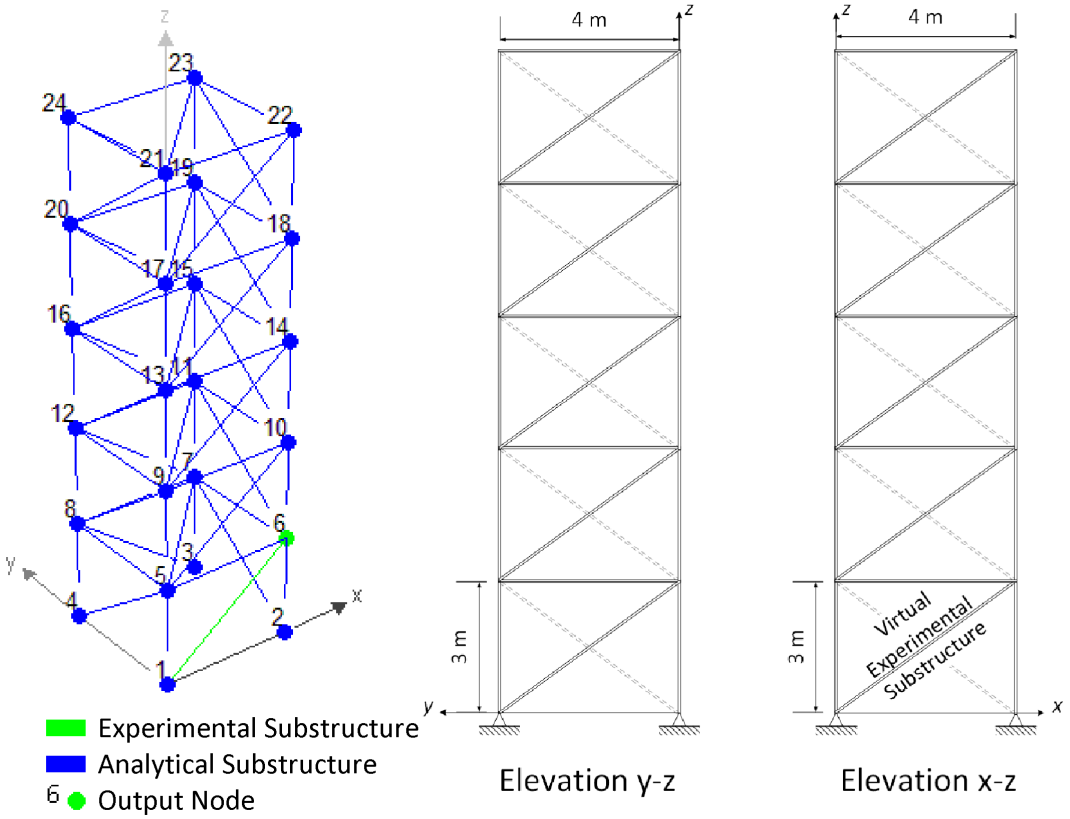



FIGURE 5.7: Analytical model of three-dimensional tower

magnify the force in the experimental member; this scaling is reflected in the figure.

Strategic selection of a structural system and applied excitation is a key aspect of effective assessment of the performance of the HSF. Large-displacement geometric effects are of interest in applications of the HSF, so the combination of structural design and excitation should be such that the resulting displacement response of a particular joint is large. To this end, modal analysis of a candidate structure and Fourier analysis of the accompanying candidate excitation were performed prior to verification, resulting in the mode shapes, natural frequencies, and relative damping factors shown in Figure 5.9. The fundamental frequency of the candidate structure preferably should be concurrent with a dominant frequency component of the excitation. From

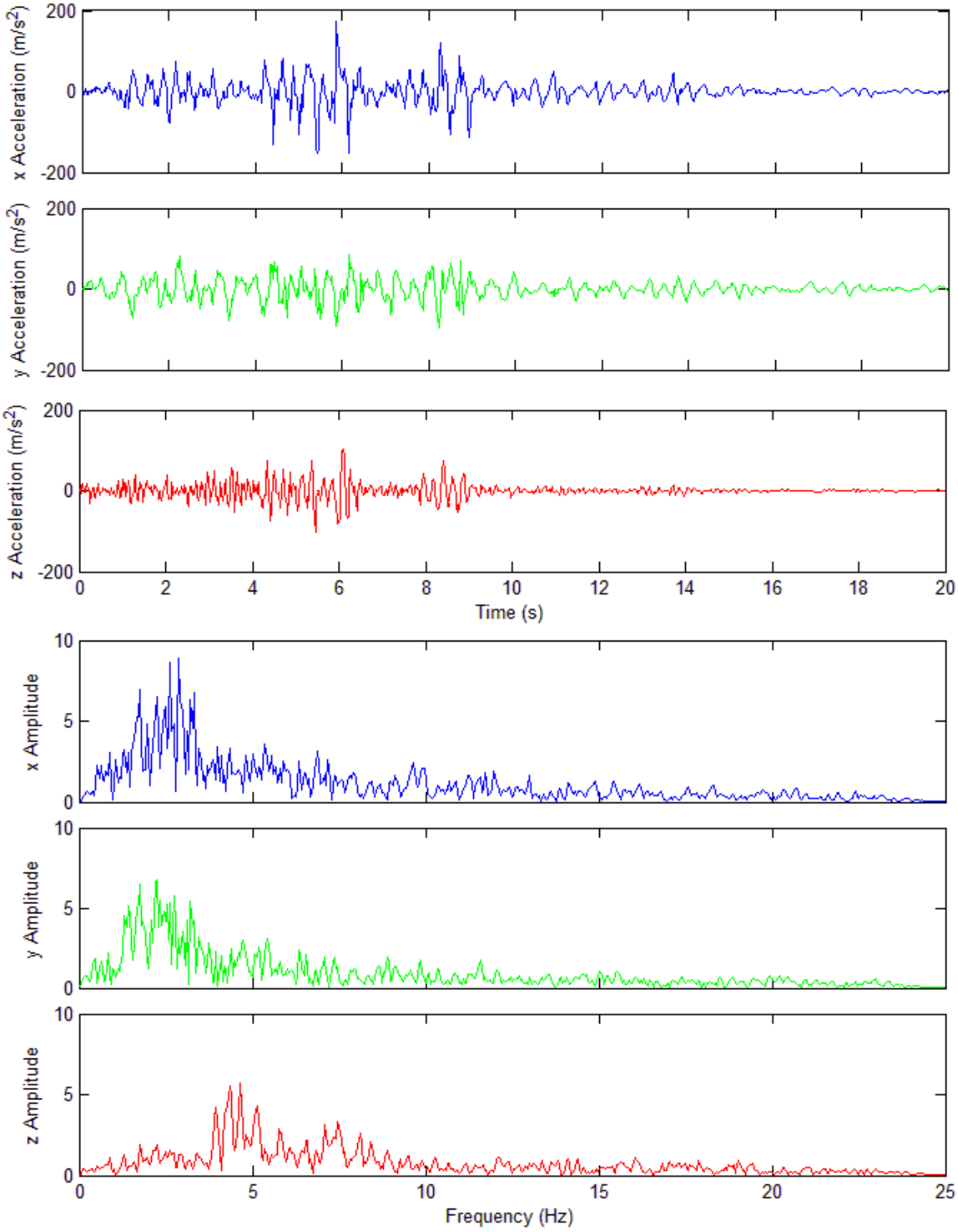



FIGURE 5.8: Northridge ground accelerations and amplitude spectra, scaled by a factor of 10

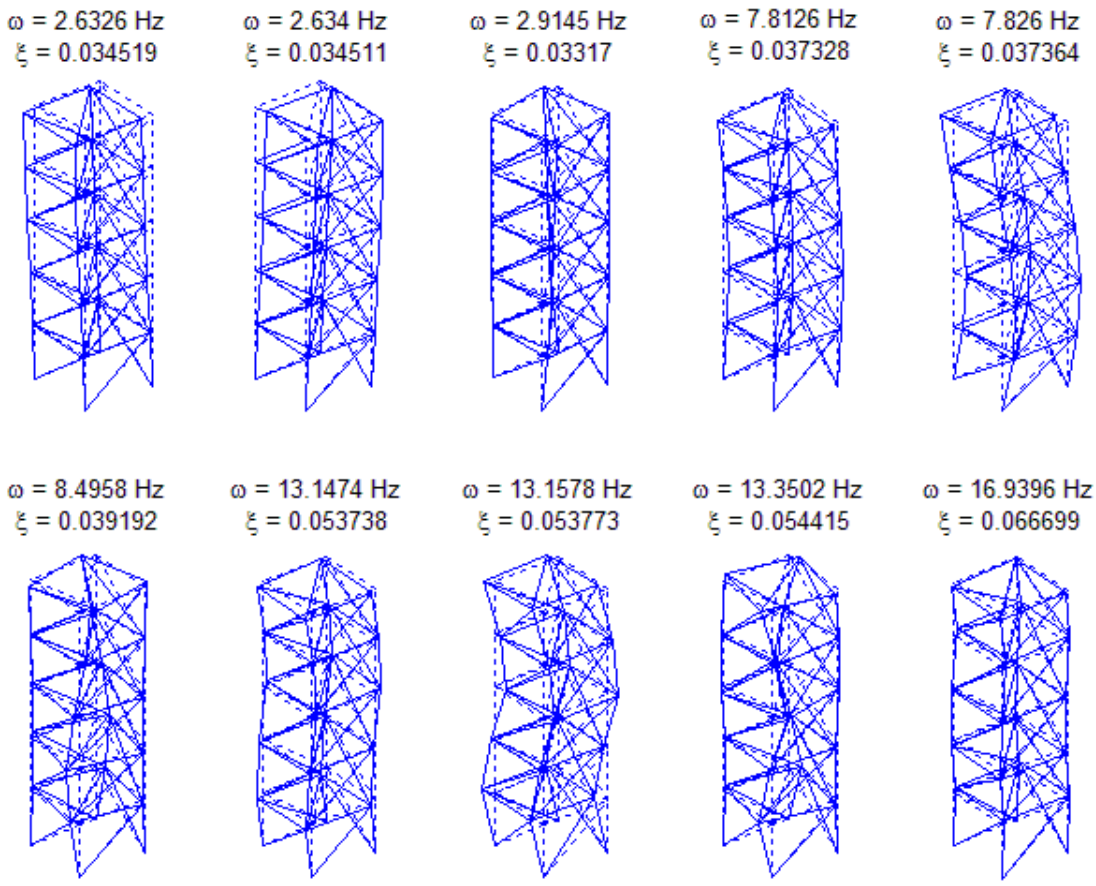



FIGURE 5.9: Mode shape plots for each of the first ten natural frequencies of the three-dimensional model

inspection of the amplitude spectra plots for the excitation, it was determined that the fundamental natural frequency of 2.63 Hz corresponds to the frequency for one of the two greatest amplitudes in the ground acceleration amplitude spectrum for the $x$ direction.

One of the diagonal braces at the lower level of the tower was used as the virtual experimental member in this verification, since the base shear developed under the ground motion should provide significant loading for that member. For the purpose of this verification, only the NLMNLG results are shown. Figure 5.10 presents the

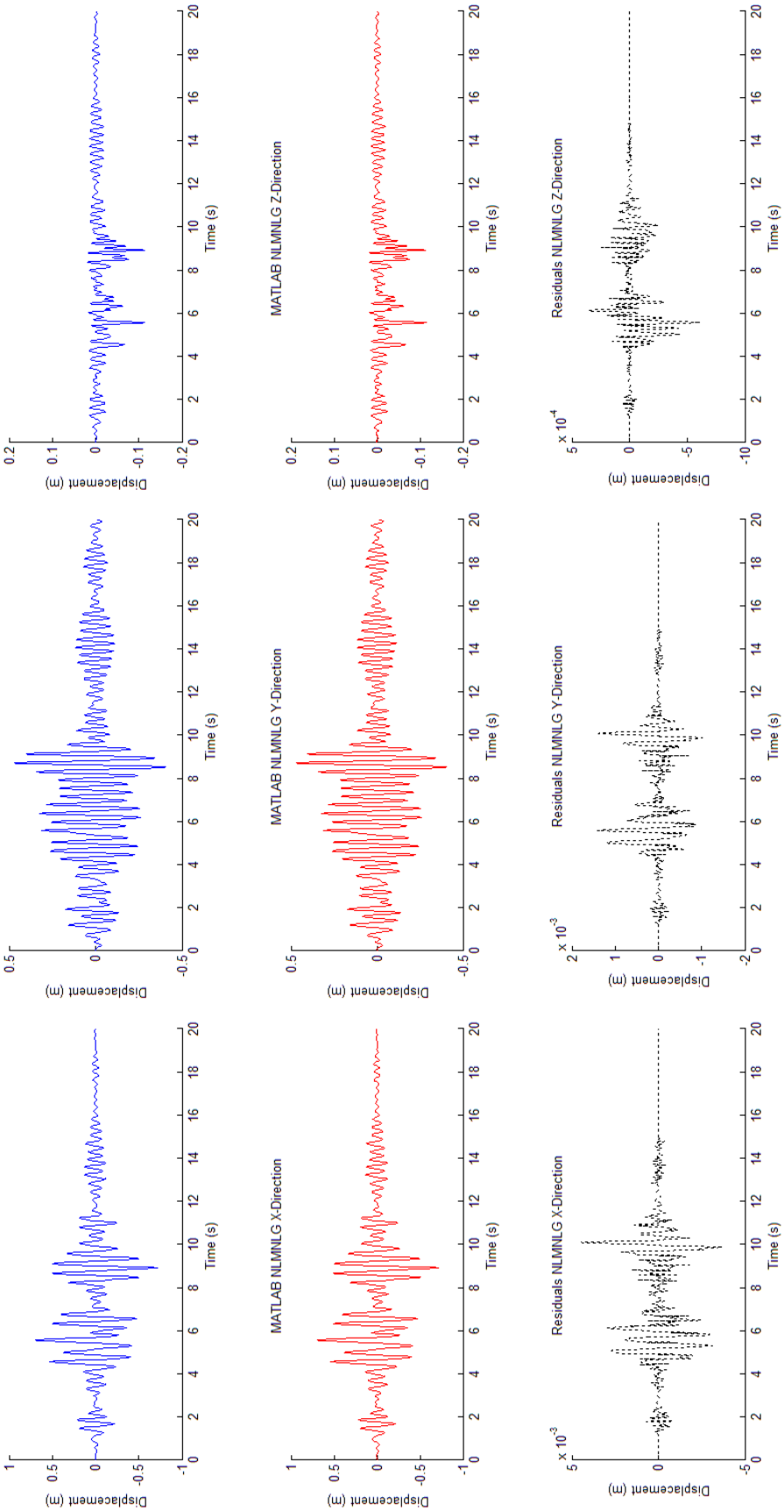



FIGURE 5.10: Verification results for base excitation of a three-dimensional structure with nonlinear material and geometric effects

SAP2000 and HSF results and residuals for the global $x$, $y$, and $z$ directions. It should be noted that the residuals appear to be largely due to a slight phase difference between the SAP2000 and HSF analyses, since the timing of the maximum residual amplitudes appears to coincide with the timing of the maximum displacement response amplitudes, particularly for the $x$ and $z$ directions. The stress-strain curve for the experimental member in the NLMNLG case is shown in Figure 5.11 and confirms that the member did experience material nonlinearity over the course of the simulation.

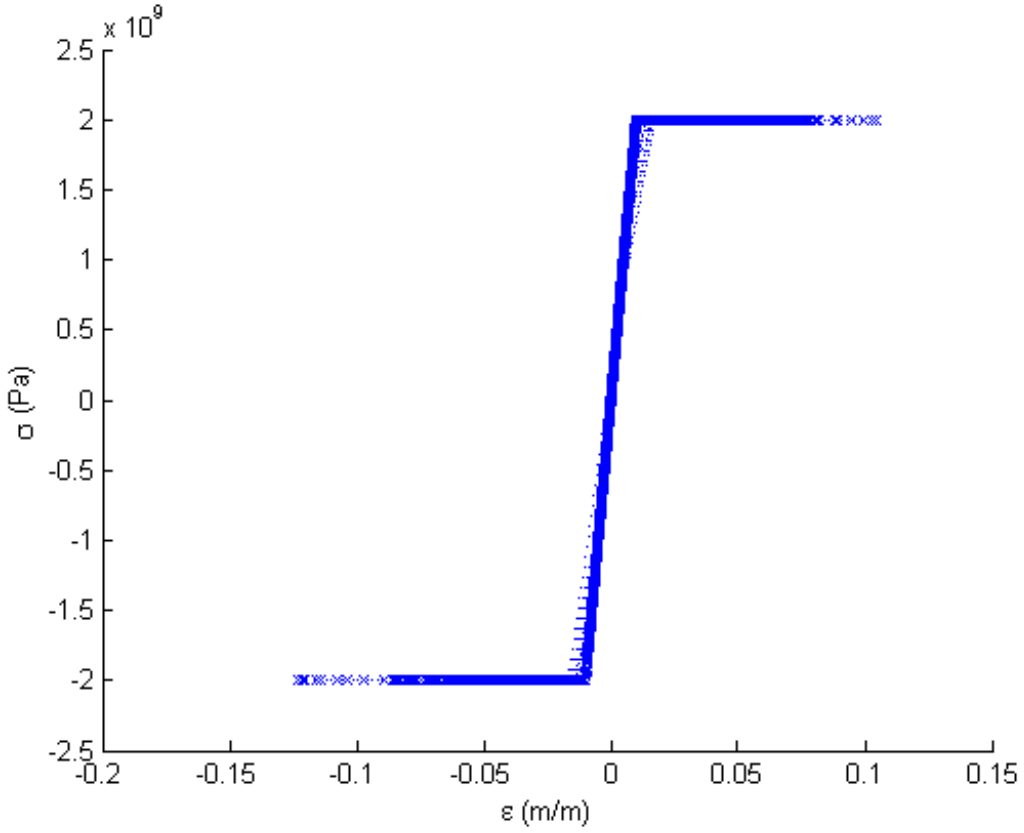



FIGURE 5.11: Stress-strain plot for experimental member in three-dimensional model

### 5.3.3 Discussion of Residuals and Source of Error in Cases with Material Nonlinearity

The primary source of error in cases exhibiting material nonlinearity is thought to be the slope discontinuity at the limit state transitions in the defined stress-strain curve. Upon inspection of Figure 5.11, it is evident that the experimental member frequently transitions rapidly between the yielded and the linear elastic segments of the stress-strain curve. In the HSF, the stiffness in such time steps is computed based on the incremental restoring force and the incremental displacement. In SAP2000, the stiffness is given instead as a direct function of the position on the stress-strain curve for the given time step. Consequently, the HSF at times computes an experimental member stiffness that is not in agreement with the exact value used by SAP2000. This is expected to be the source of the phase differences previously mentioned for the simulation involving the three-dimensional model.

To mitigate these errors, the size of the time step may be decreased, which reduces the size of the displacement increments and, thus, the likelihood that large jumps will be made across the stress-strain discontinuity. This technique was studied through the execution of multiple simulations of the verification presented in Section 5.3.1, using a variety of sampling rates. Note that the sampling rate used in the verification presented in Section 5.3.1 was originally 50 Hz, while the sampling rate used in the verification presented in Section 5.3.2 was 100 Hz. Figure 5.12 shows a plot of the maximum magnitude of the residuals for the $x$-direction displacement as sampling rate is increased (time step size decreased) for the problem presented in Section 5.3.1.

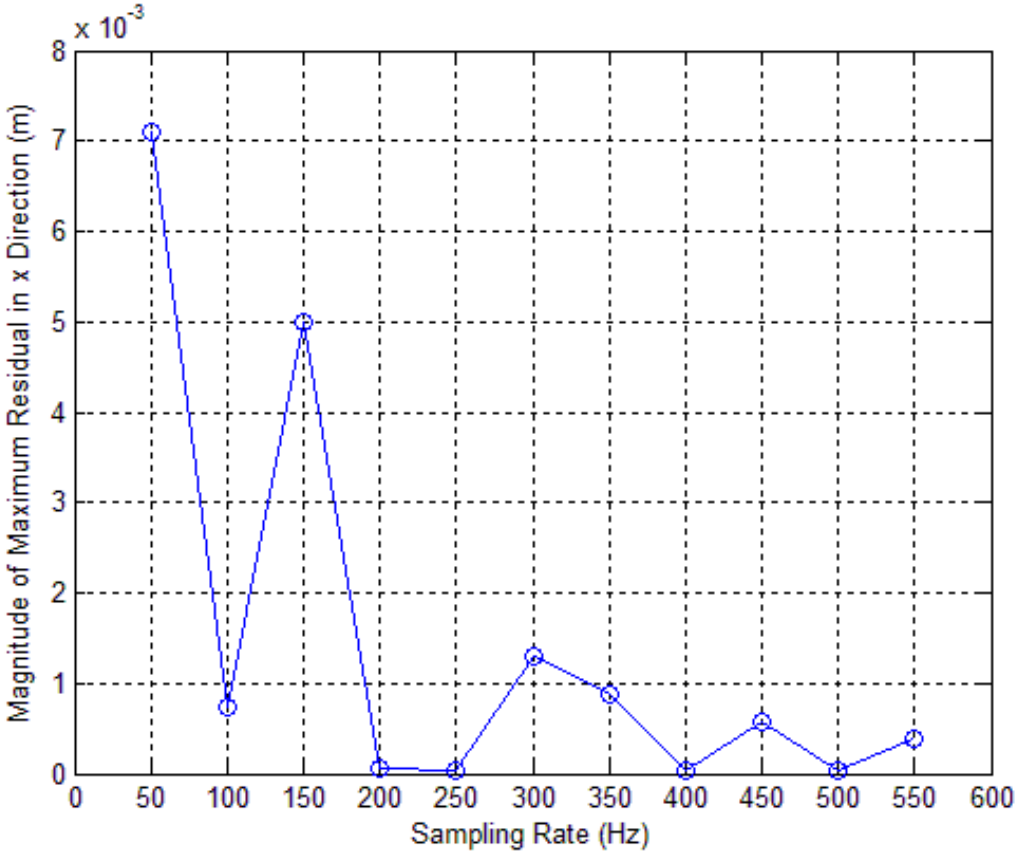



FIGURE 5.12: Maximum residual magnitude for $x$-direction displacement in the planar structure verification, as time step size decreases

Occasionally, an increase in sampling rate may appear to exacerbate the problem, since opposing phase-shift errors can sometimes offset one another. For example, at 100 Hz, the maximum residual is quite small compared to the 50 Hz case, but the residual spikes again for 150 Hz. This happens for the higher frequencies as well. Nonetheless, as sampling rate continues to increase, the maximum residuals follow a decreasing trend. Comparing the discontinuities of the stress-strain curves for the 50 Hz simulation (which has the greatest residual) and the 250 Hz simulation (which has the smallest residual) in Figure 5.13, it is evident that the 250 Hz simulation involves significantly less smoothing of the discontinuity due to displacement "jumping,"

demonstrating what is believed to be the primary reason for the different residual magnitudes of the two simulations.

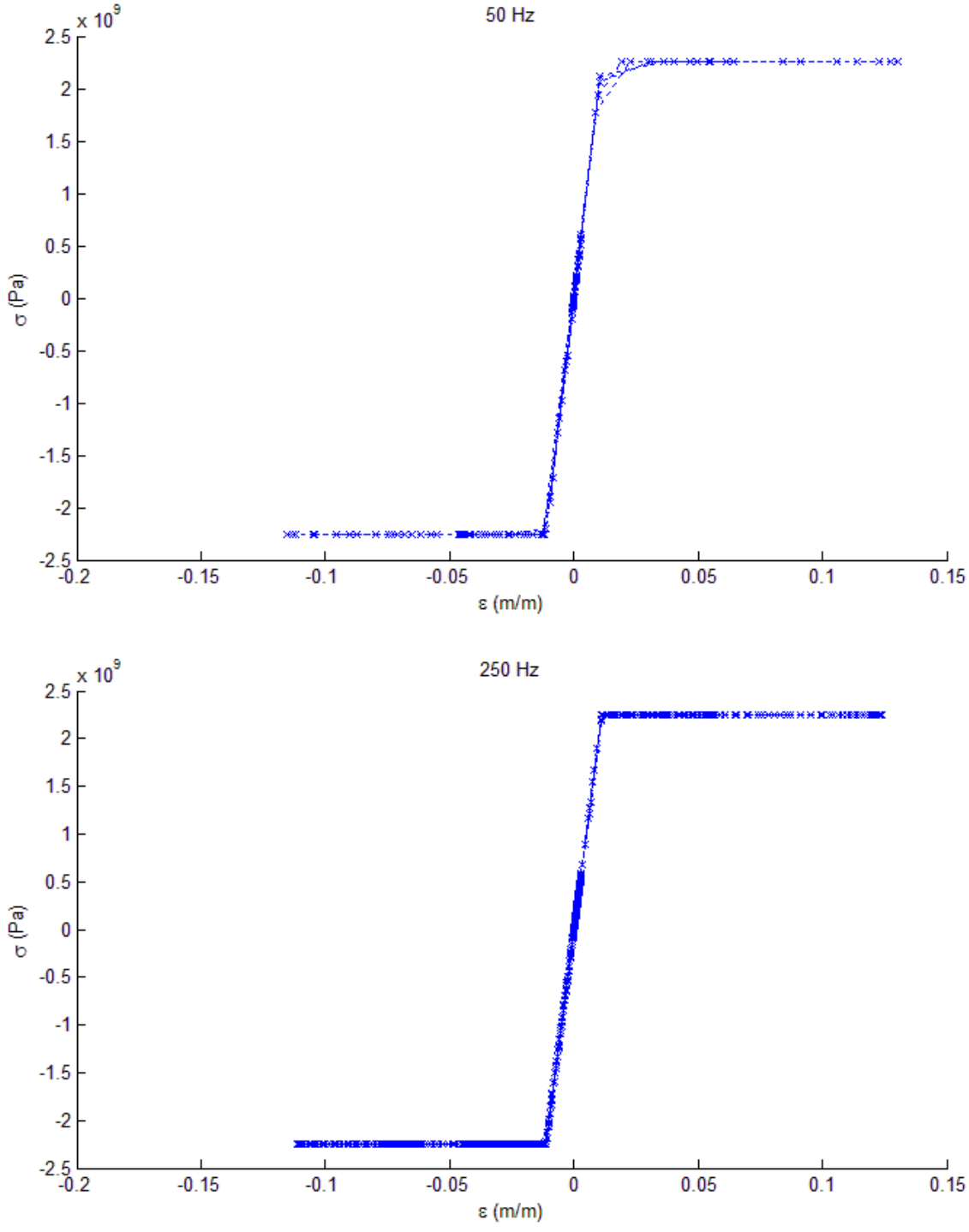



FIGURE 5.13: Stress-strain curves produced in the HSF for the 50 Hz and the 250 Hz sampling cases

# CHAPTER 6: EXPERIMENTAL VALIDATION

In the two previous chapters, the HSF was described and verified to perform satisfactorily against a benchmark commercial code across several numerical simulations. In this chapter, the implementation of the HSF in a laboratory setting is demonstrated to validate that the code is implementable, stable, and provides reasonable results compared to analytical simulations. This chapter provides a description of the hardware and interfaces used to perform the experimental validation, as well as the process used to perform the validation, a description of the simulated problem, the results of the simulation with comparison to analytical predictions, and concluding remarks and observations regarding the experimental validation.

## 6.1 Implementation of the Hybrid Testing Framework

The validation of the HSF is performed using a uniaxial MTS servohydraulic load frame with a 50-kip load cell. While the implementation is performed with this load frame, the HSF is general enough to allow implementation on any number of experimental setups with a few changes to the hardware-specific portions of the code. A schematic of the hardware and interfaces established to conduct the hybrid testing is presented in Figure 6.1 and will be discussed in this section of the thesis.

Two computers are used to interact with the MTS 793 Controller unit. The first, labeled as "MTS Computer" in Figure 6.1, is used to set up operating parameters

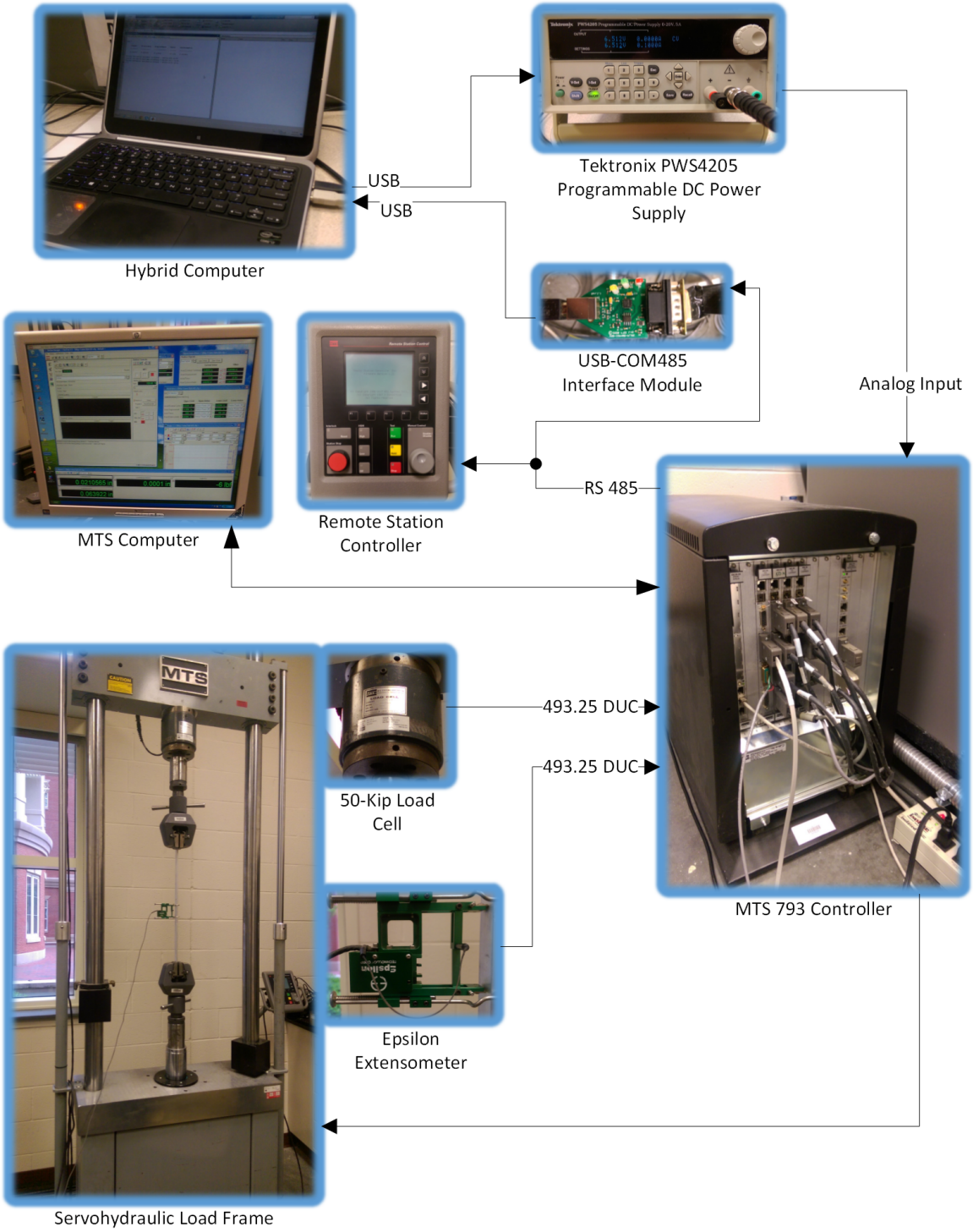



FIGURE 6.1: Schematic of hardware and interfaces used to establish the hybrid testing experimental demonstration

for the hardware used in the simulation. The second computer, labeled as "Hybrid Computer" in Figure 6.1, is used to (1) execute the HSF software routine presented in Chapter 4, (2) provide external displacement commands to the MTS 793 Controller unit, and (3) acquire force and elongation measurements from the controller. The prescribed displacement commands from the hybrid computer are provided indirectly by generating a desired voltage using a Tektronix PWS4205 Programmable DC Power Supply connected to a universal serial bus (USB) port on the hybrid computer. The voltage at each time step is assigned to the power supply using application programming interface (API) commands in the MATLAB code. The voltage from the power supply is then accepted into the MTS 793 Controller as an analog input and translated into a command displacement to be applied to the specimen by a hydraulic actuator in the load frame. A basic PIDF feedback loop between the MTS 793 Controller and an external extensometer (Epsilon 3542-0200-020-ST, gage length 2.00 in.) provides closed-loop control between the actuator motion and command displacement. Elongation of the specimen is monitored with the extensometer, while the restoring force is measured using the 50-kip load cell in the MTS load frame.

It should be noted that the elongation measured by the extensometer is used in this setup in lieu of displacements measured by the linear variable differential transformer (LVDT) in the MTS load frame. This is due to the fact that the wedge grips used to restrain the ends of the specimen were observed to perform unreliably, meaning that the LVDT measurements correspond to the displacement of the actuator rather than the specimen itself. As a result, LVDT measurements suggested a modulus of elasticity that was significantly lower than expected, as well as apparent hysteresis during

cycling of tensile stresses well-below the anticipated yield stress of the specimen. This issue was remedied by using extensometer readings at the midspan of the specimen to directly measure the strain. These measurements are scaled in order to project the true displacement of the specimen.

As measurements are collected, the analog signals from the extensometer and the load cell are directed into 493.25 digital universal conditioner (DUC) channels in the MTS 793 Controller, where they are converted to digital form using calibration factors prescribed in the MTS software. To efficiently pass these digital measurements from the controller to the hybrid simulation computer, a connection is established at the RS 485 interface that is commonly used to communicate between the MTS 793 Controller and the Remote Station Controller. The digital measurement signal is converted in-line with a USB-COM485 interface module between the controller and the USB port of the hybrid computer, where measurement data is read into the HSF.

The following is a general procedure for executing hybrid simulations with this hardware setup:

1. Auto-tune the MTS software to prescribe optimal gain settings for the extensometer PIDF loop.

2. Define an External Command Procedure in MultiPurpose TestWare® (MPT) to prepare the MTS system for hybrid computer control.

3. Establish limit detectors for force, displacement, elongation, and external command to prevent unwanted damage to the specimen or hardware.

4. Run the setup portion of the HSF on the hybrid computer until prompted to

zero measurement signals.

5. Apply offset corrections to zero the force, displacement, elongation, and external command measurements of the MTS system.

6. Run the MTS MPT program to set the control mode to external command.

7. Resume MATLAB, allowing it to drive the simulation through the closed-loop communication interface until the completion of the simulation.

8. By stopping the MPT program, disconnect the controller from the external command and return the actuator to displacement control.

## 6.2 Problem Description and Setup

The structural model used for the experimental validation represents a power transmission tower subjected to a hypothetical dynamic load (specified as $P$) of a single galloping conductor wire on one arm of the tower and a galloping ground wire attached at the top of the tower, as shown in Figure 6.2. The frequency of the excitation due to galloping was chosen to correspond with the fundamental frequency of the tower model. The structure was modeled using structural steel having a Young's modulus of 29,000 ksi and unit weight 490 pcf. The tension element supporting the conductor at the end of the cross arm was selected as the experimental member (or experimental substructure) for this validation, as it is pre-tensioned under the static dead load of the conductor wire. Due to experimental limitations and ease of implementation, the tension element serving as the experimental member was physically modeled with a 1"$\times$0.25" bar of aluminum flat stock.

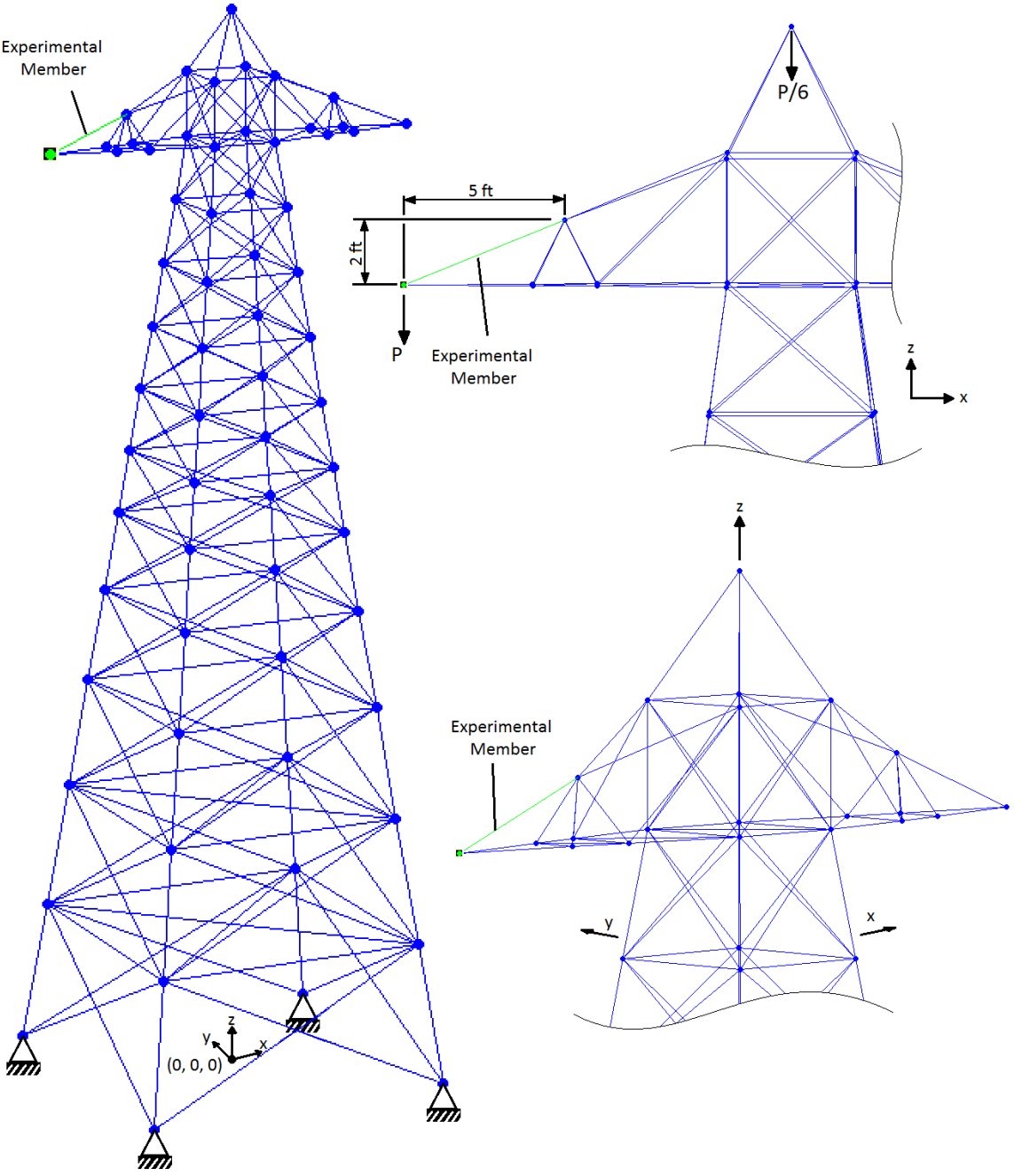



FIGURE 6.2: Transmission tower model subjected to hypothetical galloping conductor and ground wire for experimental validation

Prior to the experimental validation, a preliminary nonlinear direct time-history analysis was performed to conservatively determine the maximum expected displacement for the specimen during the experiment. In order to establish a conservative estimate of the maximum displacement, the modulus of elasticity for aluminum was intentionally selected to be a soft initial estimate (9,750 ksi) by comparison to the commonly-observed 10,600 ksi. The maximum displacement estimate is important for scaling the voltage signal used for prescribing the external command and for setting limit detectors to power down the hardware under unsafe conditions. The stiffness of the experimental member is updated during the simulation, so the assumed modulus of elasticity of the experimental member is inconsequential in the actual experiment. The experimental member was assigned a unit weight of 169 pcf in the model.

It is strongly emphasized that the tower model developed for the validation is not meant to provide a realistic representation of an actual transmission tower design. It is intended to serve as a tool for demonstrating the implementation of the HSF in the context of a large-scale space truss in the targeted application area for the HSF. In its current state, the HSF only supports analytical substructures with axial elements assumed not to buckle under compression loads or yield under tension loads. In other words, failures are considered only in the experimental substructure. Since transmission towers have often been observed to fail below the waist due to global or local buckling of bracing members [Albermani et al., 2009, Rao et al., 2010, Rao et al., 2012], the selection of a member of the cross arm as the experimental member (and, thus, the critical member) may not be realistic. This aspect of the development of the tower model was neglected, since the full-scale compression failure of a long member

of the tower is difficult to realistically simulate given the hardware selected for the experimental validation of the HSF. Since the available span for the servohydraulic load frame limited the length of the specimen, the specimen was scaled to approximately forty percent of the total length of the experimental member of the model. Such scaling was only permissible under the condition that the unbraced length of the experimental member in the model was not a factor, so an experimental member was chosen that would experience only tension. The specimen was installed in the load frame using wedge grips at the ends, so the controlling failure mode for the specimen was assumed to be gross section yielding. Additionally, the dynamic properties of the member were accounted for numerically, since the simulation was performed pseudodynamically. As a result, the physically-represented length of the specimen was not a factor in the analysis itself.

Although the transmission tower model was not explicitly developed to provide an accurate representation of actual transmission tower designs, it was configured deliberately to have a realistic fundamental natural frequency, vibration modes, and relative damping factors. According to Appendix F of the ASCE *Guidelines for Electrical Transmission Line Structural Loading*, latticed towers commonly exhibit a fundamental frequency of 2.0 to 4.0 Hz, with damping ratios of approximately 0.04 [Wong and Miller, 2009, p. 115]. Geometric and cross-sectional properties of the tower design were adjusted to identify a representative dynamic model according to this guidance. Several vibration modes of the tower are provided in Figure 6.3, along with corresponding frequencies and damping ratios.

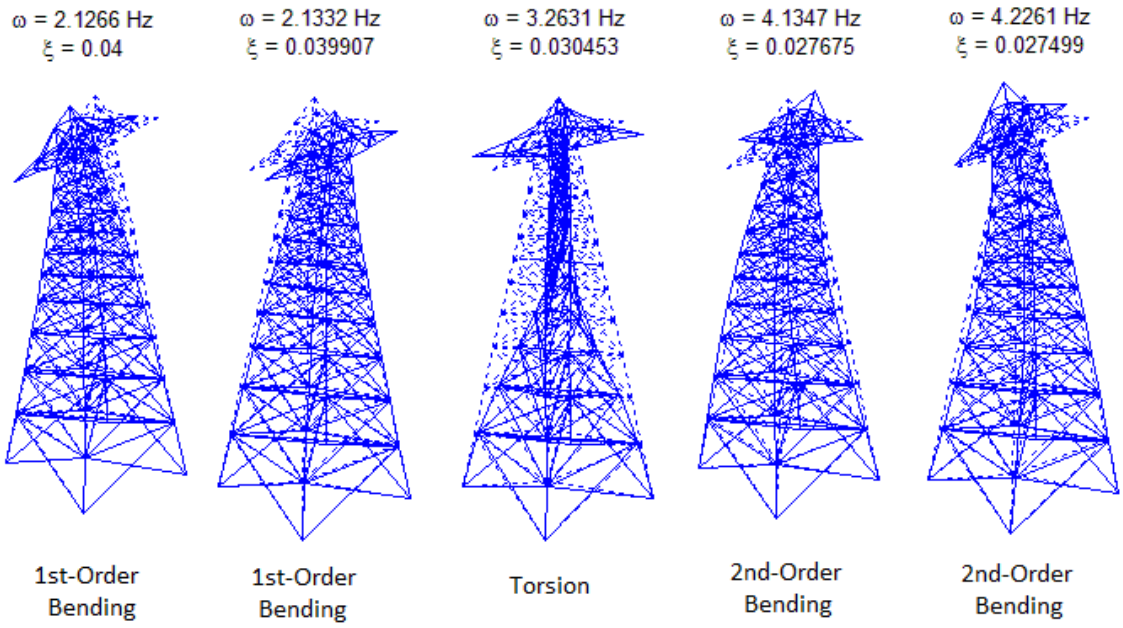



FIGURE 6.3: First five vibration modes for transmission tower model

In addition to providing approximate fundamental frequencies and damping ratios for common latticed towers, ASCE cites an Electric Power Research Institute (EPRI) journal that reports common oscillation frequencies for conductor galloping ranging between 0.08 and 3 Hz [Rawlins et al., 1979]. Since the fundamental frequency of the tower model was computed as 2.1266 Hz, it was possible to select a coinciding frequency within that range of conductor galloping frequencies for the excitation signal. Although the frequency of oscillation is representative of common conductor galloping loads, it should be noted that the magnitude of the excitation is not: the load magnitudes have been selected for the sake of producing desired tensile stresses in the experimental member as well as notable geometric effects for the arm of the tower model. The magnitude of the load on the ground wire is selected to be approximately one-sixth of the load on the conductor, for convenience of demonstration and loosely

based on the approximate relative magnitudes of the conductor and ground wire static dead loads used in a design example problem [Wong and Miller, 2009].

Two validations were performed. The first validation was performed using excitations chosen to generate a broad range of axial stress magnitudes in the experimental member without causing it to yield. The load $P$ for the first validation is shown in Figure 6.4. In this hybrid simulation, the static conductor dead load is taken as 0.9 kips, and the maximum magnitude of the load due to conductor galloping is taken as 0.63 kips. Thus, the load $P$ is increased gradually from zero to 0.9 kips at the start of the simulation, as shown in Figure 6.4. $P$ is then held at 0.9 kips to allow an induced dynamic excitation of the structure to subside. After this initial phase of loading, the "galloping" initiates, as $P$ oscillates at 2.1266 Hz with increasing amplitude until the halfway point of the simulation. Beyond the halfway point of the simulation, the oscillatory loading begins decreasing in amplitude. When the load oscillations subside, the static dead load is held again. Finally, the structure is unloaded to allow

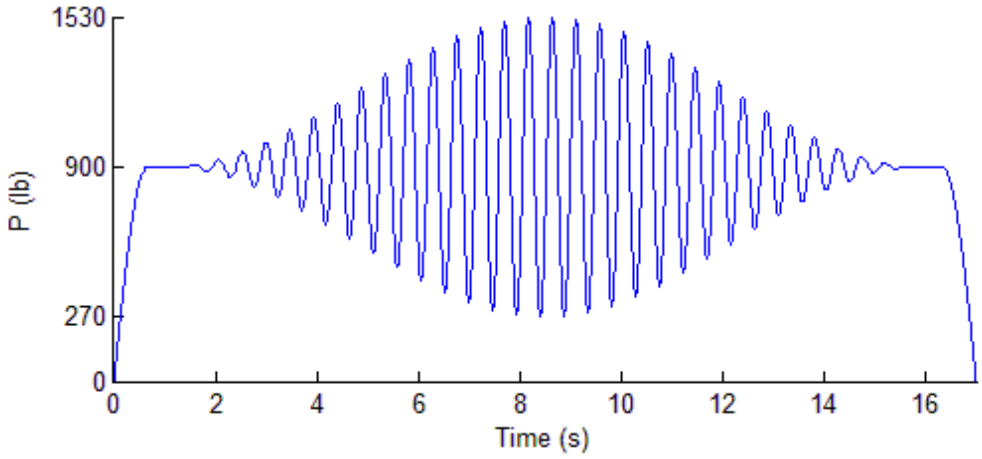



FIGURE 6.4: Load $P$ applied to transmission tower model for linear elastic simulation

for measurement of any residual strain in the specimen and to permit removal of the specimen from the load frame.

The second validation was performed using excitations designed to produce significant material nonlinearity in the specimen. The load $P$ for the second validation is shown as a function of simulation time in Figure 6.5. This load $P$ is similar to that of the first validation. Here, the static dead load of the conductor is taken as 1.955 kips, and the maximum magnitude of the load due to conductor galloping is taken as 0.805 kips. The loading procedure is otherwise identical to that of the first validation.

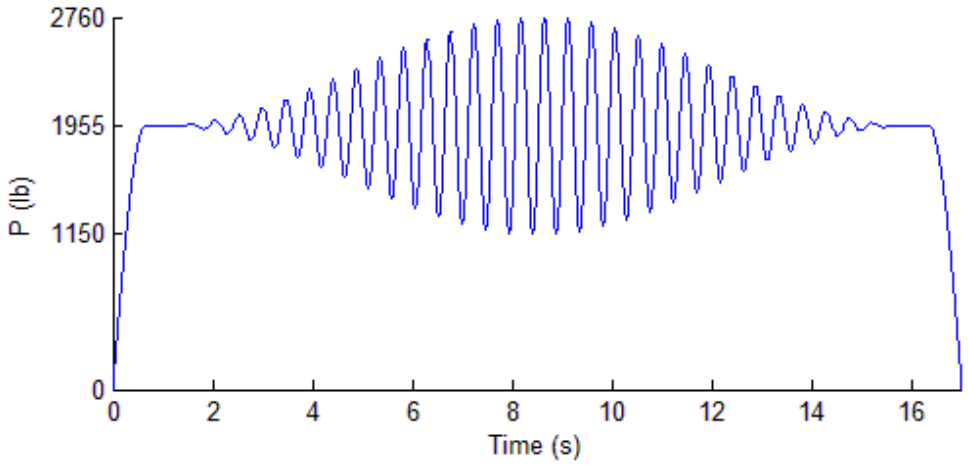



FIGURE 6.5: Load $P$ applied to transmission tower model for nonlinear simulation

## 6.3 Experimental Results and Observations

The initial hybrid simulation performed in the linear elastic range of the material response produced results consistent with those from a SAP2000 nonlinear direct time-history analysis of the full model with the experimentally-observed modulus of elasticity assigned to the experimental member. The stress-strain curves from the experiment and the subsequent SAP2000 analysis are shown in Figure 6.6. The

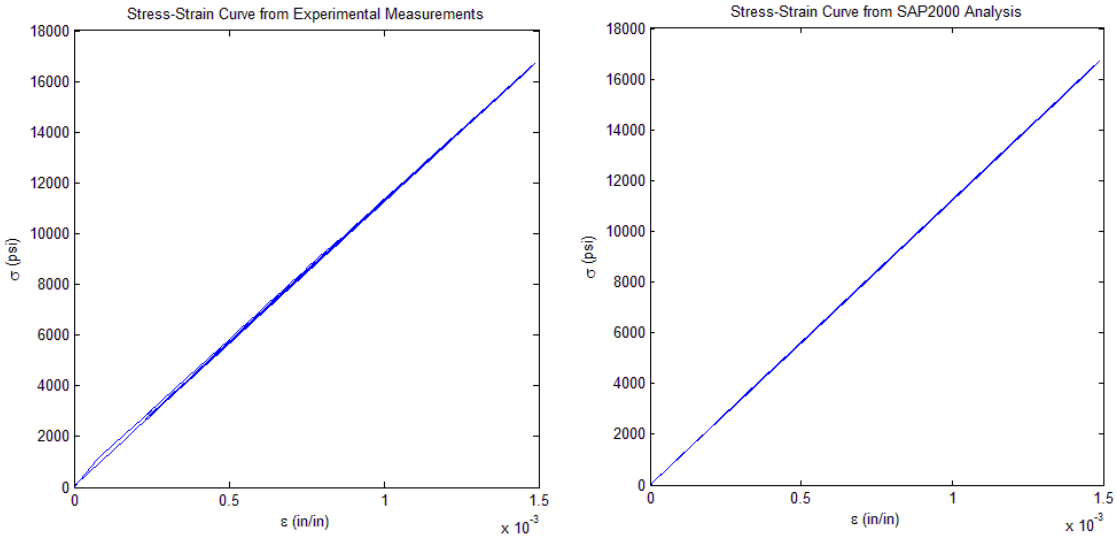



FIGURE 6.6: Experimentally-observed stress-strain relationship and analytically-simulated stress-strain relationship

stress-strain curve from the experiment indicates minor nonlinearity in the specimen, which may account for some of the discrepancy between the experimental simulation and the finite element analysis.

The displacement time histories of the node at the tip of the loaded arm are presented for the $x$, $y$, and $z$ directions in Figure 6.7. These plots include the experimental results, SAP2000 analysis predictons, and the residuals between the results and the predictions. As shown in the figure, the agreement is strong with residuals several orders of magnitude less than the peak displacements.

For the nonlinear simulation, the specimen was initially yielded under the application of the pre-tension applied as the static dead load. As a result, the oscillations produced by the galloping of the conductor generated plastic deformation immediately. By the conclusion of the simulation, the specimen experienced approximately 0.243 percent

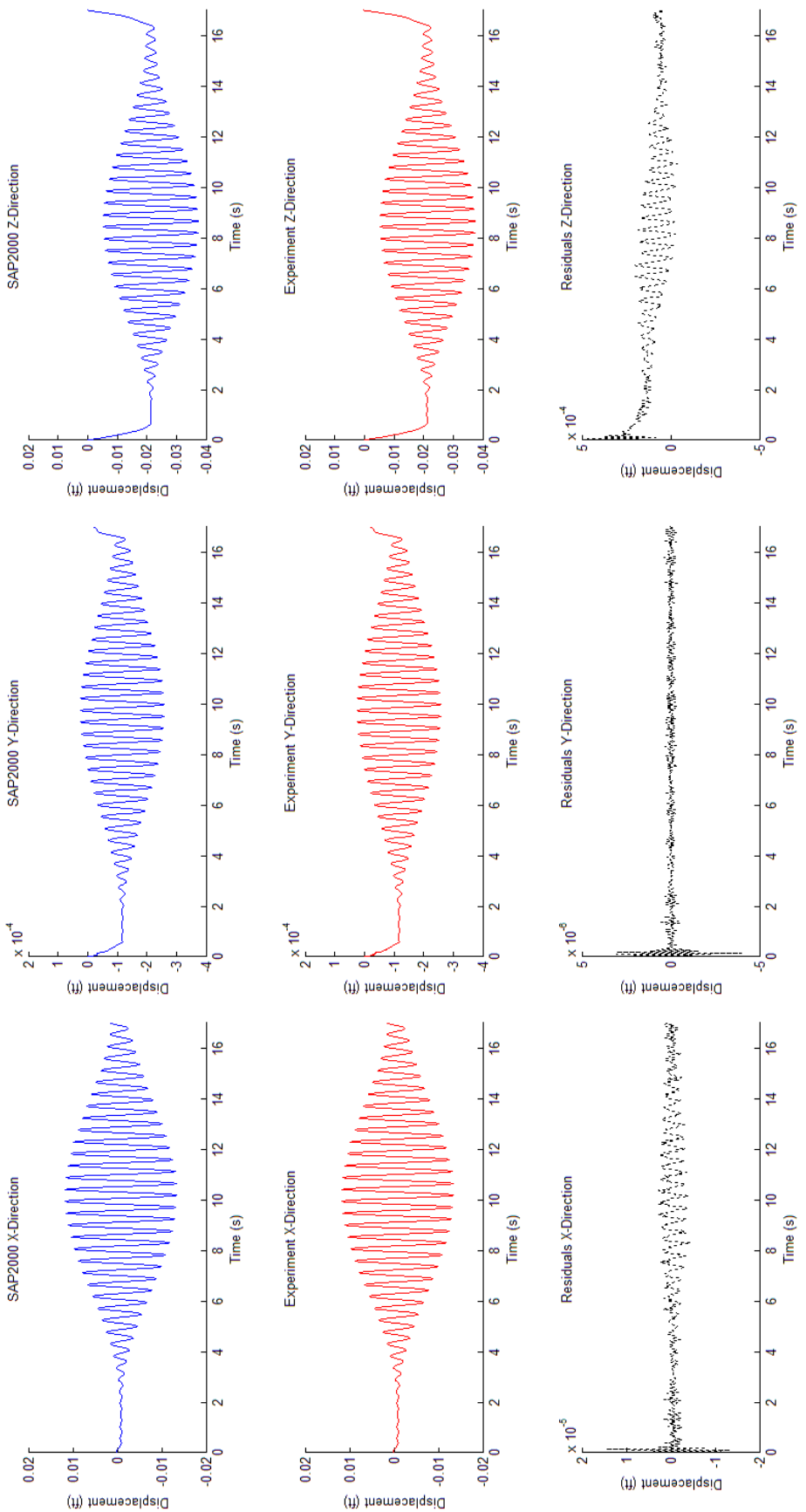



FIGURE 6.7: Displacement time histories for simulation in the linear elastic region

permanent set strain. This corresponds to approximately 0.157 inches of elongation for the full-length experimental member. For the *ex post facto* SAP2000 nonlinear direct time-history analysis performed for comparison against the experimental results, a nonlinear material model was approximated using six points, as shown in Figure 6.8.

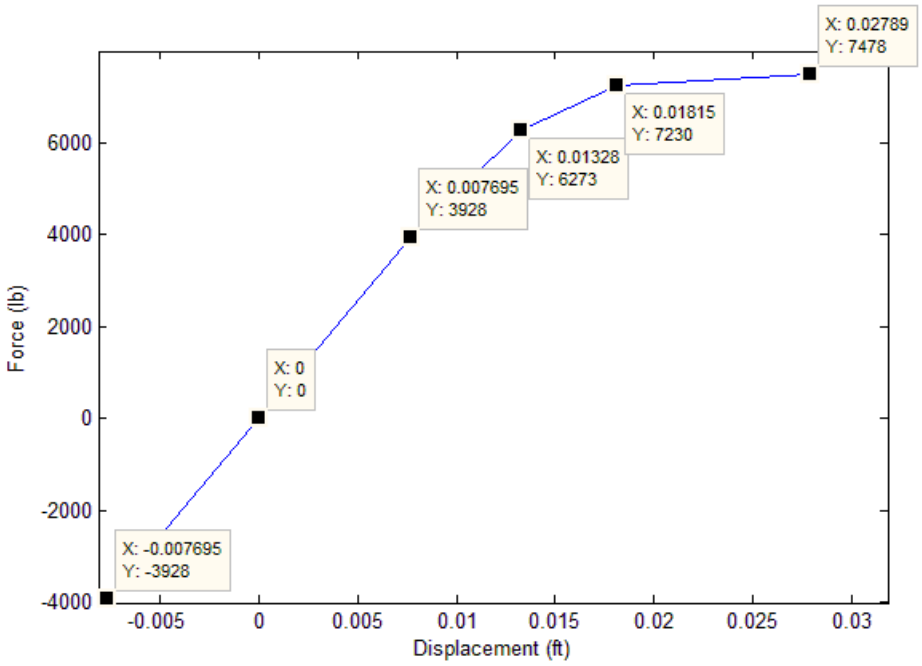



FIGURE 6.8: Force-displacement relationship definition used in SAP2000 to perform *ex post facto* analysis for comparison to experimental nonlinear simulation

These points were determined based on the envelope of the stress-strain curve generated from the experiment, with the exception of the point in the compression region (assigned to satisfy the requirement of at least one positive and one negative point in the material model definition for SAP2000). The SAP2000 analysis was performed with a kinematic hysteresis model. The stress-strain curve generated during the experiment is shown in Figure 6.9 alongside the stress-strain curve produced from the subsequent SAP2000 analysis.

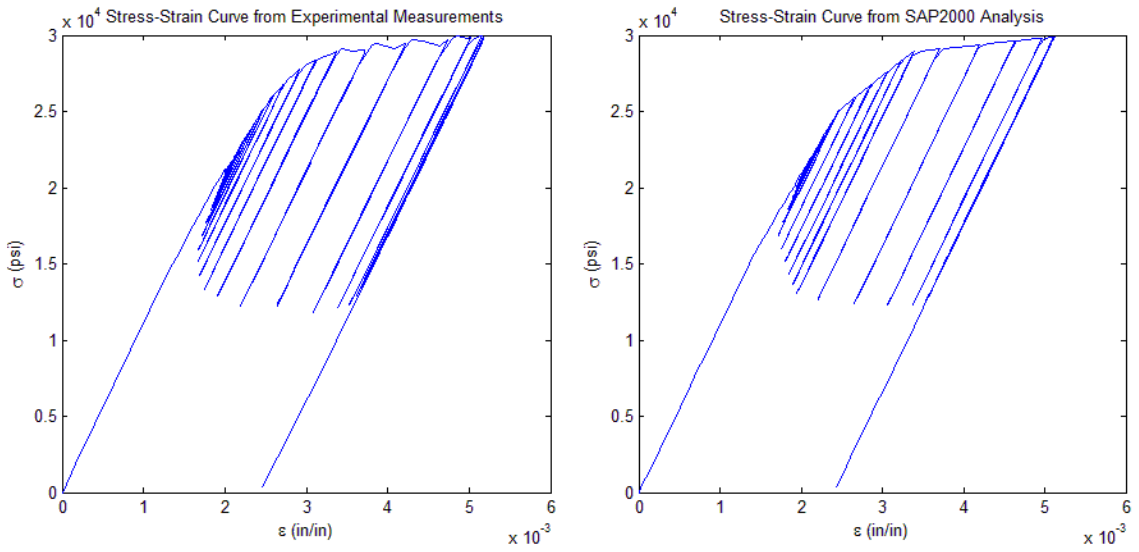



FIGURE 6.9: Experimentally-observed hysteresis and analytically-simulated hysteresis

The hysteresis exhibited in the hybrid simulation and the nonlinear finite element analysis correlate strongly. Additionally, the peak dynamic strain exhibited by the experimental member and the permanent set strain are in strong agreement. The strength of the correlation between the experimental hybrid simulation and the finite element analysis serve to validate the HSF and hardware implementation presented in this thesis.

The displacement time history of the node at the tip of the loaded arm is presented for the $x$, $y$, and $z$ directions in Figure 6.10. These plots include the experimental results, SAP2000 analysis predictions, and the residuals between the results and the predictions. The displacement of the node at the tip of the loaded arm for the nonlinear simulation was in close agreement to the corresponding SAP2000 analysis, though the magnitude of the residuals relative to the peak displacements were larger than that of the initial validation performed in the linear elastic range of the material. For

this second validation, residuals were generally less than 3 percent of the magnitude of the peak displacements. These residuals are small enough to be explained by the difference between the approximate material model defined for the SAP2000 analysis and the realistic stress-strain envelope exhibited by the experimental member. It should be noted that both simulations performed for validation took into consideration the large-displacement geometric nonlinearities.

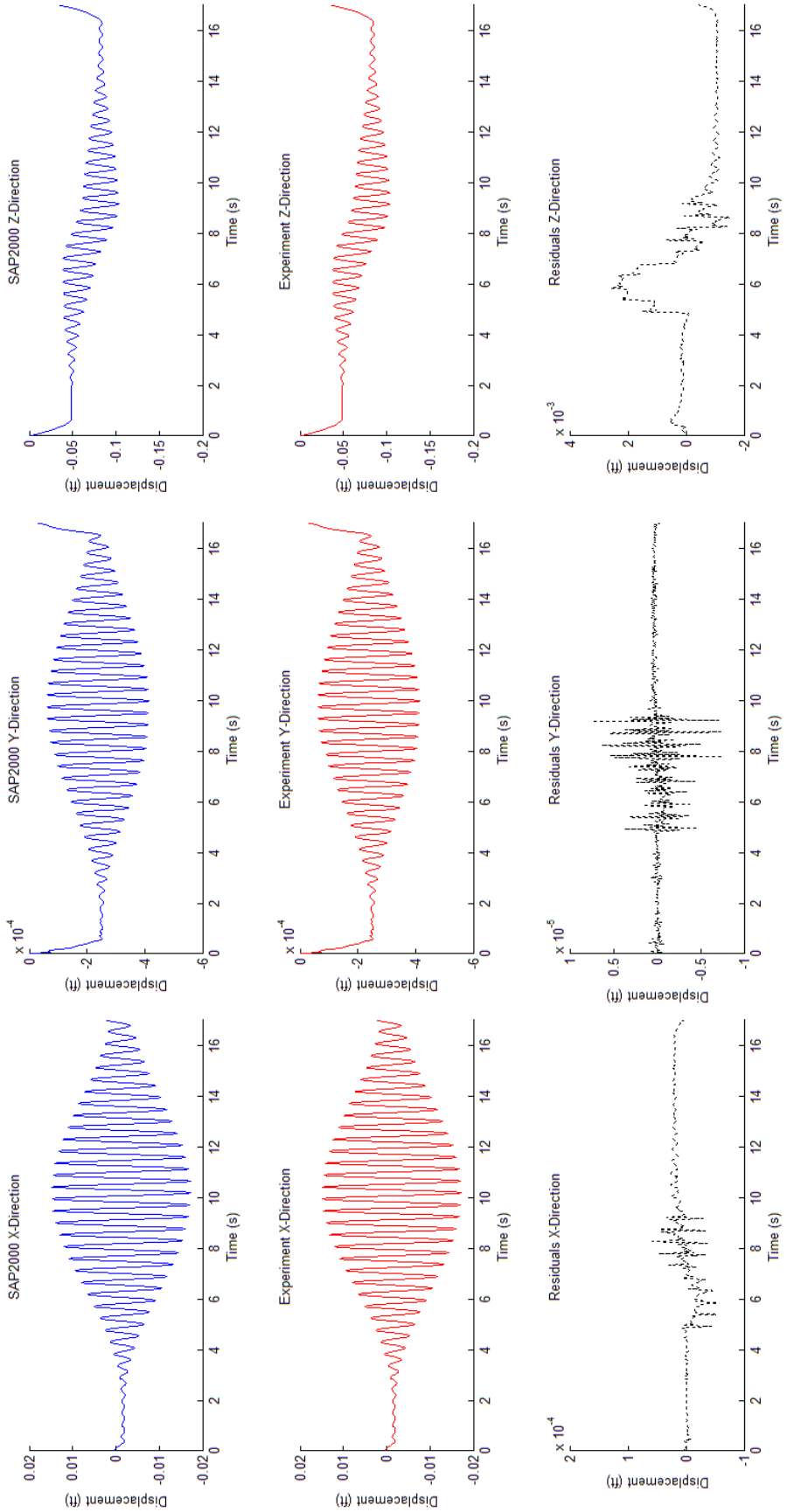



FIGURE 6.10: Displacement time histories for nonlinear simulation

# CHAPTER 7: CONCLUSION

## 7.1 Summary

Hybrid simulation has become increasingly popular in recent years, due to its ability to produce accurate experimental simulations without sacrificing economy or efficiency. In order for the engineering community to benefit fully from the potential that hybrid testing offers, hybrid simulation software frameworks must be implemented in a greater number of laboratories. In this thesis, a software framework for hybrid testing of geometrically nonlinear latticed structures was presented. This software framework performs pseudodynamic hybrid simulation with any experimental substructure that can be interfaced with the corresponding analytical substructure with a single degree of freedom. An iterative method proposed by Ahmadizadeh [Ahmadizadeh, 2007] is used to solve the discrete-time dynamic equation of motion by numerically iterating on the entire structural system, using extrapolation of a polynomial-based force-displacement relation for the restoring force provided by the experimental substructure to overcome the challenge of avoiding iterative imposition of physical displacements on the specimen.

The verification process presented in Chapter 5 detailed a method for reliably assessing the proper development and numerical implementation of a hybrid simulation software framework using comparative analysis to a commercial finite element

package, such as SAP2000. The resulting verification indicated that the developed software routine performs very accurately in comparison to nonlinear direct time-history integration in the SAP2000 environment. Both planar and space trusses with harmonic and ground acceleration excitations were analyzed for the verification, which was confirmed to exhibit residuals several orders of magnitude smaller than the displacements experienced by the models. Following verification of the numerical implementation, an experimental hybrid test was conducted to demonstrate the physical application and validate the experimental implementation by comparison to finite element analysis. One validation was conducted for linear elastic response of the member, and a second extended the member response to nonlinear plastic behavior. The validations, presented in Chapter 6, demonstrated successful implementation of the hybrid simulation software in the context of a hypothetical power transmission tower subjected to dynamic loading from a galloping conductor wire and a galloping ground wire. The validation also further confirmed the accuracy of the software routine, as an *ex post facto* SAP2000 analysis (with the material properties observed in the hybrid simulation assigned to the model) produced results with strong agreement to those of the experiment.

In its current form, the software framework presented in this thesis is capable of facilitating hybrid pseudodynamic simulations of latticed structures subjected to base excitation and time-varying nodal loads. Limited research has been done in the area of hybrid simulation with power transmission structures or space trusses in general. The software framework presented in this thesis may be a powerful tool to enable future research in this area, including but not limited to the study of galloping conductors.

## 7.2 Recommendations for Future Work

The software framework presented in this thesis is well-suited for slow pseudodynamic hybrid simulation with trusses exhibiting significant geometric nonlinearity, as well as material nonlinearity in the experimental substructure. In order to expand the capabilities of this framework, future research efforts should be aimed toward the implementation of several additional facets for improved operation and analysis capabilities. These items are described in the following subsections.

### 7.2.1 Extension of the Analytical Solver to Include Frame Elements

In order to extend the software framework presented in this thesis into structures with frame elements, the global stiffness and mass matrix assemblies must be expanded to include rotational degrees of freedom. Additionally, member force computations will no longer be limited to axial forces. It is recommended that the force recovery formulation, presented in Appendix B of *Matrix Structural Analysis* [McGuire et al., 2000], be used to account for natural deformations of the elements while excluding rigid body movements of the elements in computing the member forces. Transformation matrix formulation will also need to be expanded, as well as the model definition input structure provided in input files.

### 7.2.2 Multiple-Degree-of-Freedom Interface Between Experimental and Analytical Substructures

Currently, the developed software framework is limited to applications in which the experimental substructure can be considered to interface with the analytical

substructure across a single degree of freedom, as the experimental substructure stiffness is represented as a scalar computed as the quotient of the change in measured restoring force and the change in measured displacement. Ahmadizadeh provides insight into methods of generating updated tangent stiffness matrices for experimental substructures that interface with the analytical substructure over multiple degrees of freedom [Ahmadizadeh, 2007, pp. 196-206]. A related improvement to the experimental substructure capabilities would be to include multiple experimental substructures. This would be necessary to implement geographically-distributed hybrid simulation.

### 7.2.3 Numerical Damping

In the course of the verification process, slight discrepancies were observed between results produced from the software framework and those from SAP2000 for cases where the numerical damping parameter ($\alpha$) of the numerical integration scheme was assigned as nonzero. The results of SAP2000 did not reflect noticeable variation for $\alpha \in [-\frac{1}{3}, 0]$ within the case of base excitation of the three-dimensional tower, but the results from the software framework did. This suggests a potential misuse of the numerical damping parameter in the software framework. Combescure and Pegon state that "if the loss of stiffness implies also a large shift in frequency of the high frequency modes, the $\alpha$-damping does not work any more and the I-modification tends to aggravate the situation" [Combescure and Pegon, 1997]. It is recommended that further research be conducted regarding the effects of non-zero $\alpha$ prior to utilizing the software framework with such values for $\alpha$.

### 7.2.4 Geometric Stiffness Matrix

In the process of verifying the software framework presented in this thesis, it was determined that the use of the element geometric stiffness matrices, as described in Section 3.1, produced results from the software framework that were inconsistent with those of SAP2000 analyses. Although the formulation presented in Section 3.1 is trusted as a correct mathematical treatment of geometric nonlinearity, the software framework currently operates on the basis of a stiffness matrix formed using

$$\mathbf{k} = \frac{EA + f_x}{L} \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

rather than Equation 3.1 to determine the local stiffness matrix for each member. The above $\mathbf{k}$ formulation has been observed to produce results that closely agree with those produced by the SAP2000 nonlinear direct time-history analysis. While these results are satisfactory, the source of the discrepancy in the originally-planned (Equation 3.1) formulation should be identified and corrected in future research. This must occur prior to the extension of the software framework to frame element applications and before introduction of more complex geometric nonlinearities.

### 7.2.5 User Interface

For the software framework presented in this thesis, the numerous dialog boxes used for designating analysis options and user inputs can be improved upon to overcome potential drawbacks inherent to the current user interface. First, the current interface may result in a slightly disjointed or disorienting setup process, unless the user has already developed a strong familiarity with the setup process, the principles of structural dynamics, and hybrid testing as a whole. Secondly, with so many prompts required to initiate the analysis, it can be easy to enter a wrong value or click the wrong button, and mistakes can only be corrected by restarting the entire process. Due to these two issues, it is recommended that a more intuitive user interface be created using the Graphical User Interface Design Environment (GUIDE) in MATLAB for future revisions of the software framework. This would be an effective way to consolidate and organize the dialog box prompts into a user-friendly form.

# REFERENCES

[Ahmadizadeh, 2007] Ahmadizadeh, M. (2007). *Real-time seismic hybrid simulation procedures for reliable structural performance testing*. ProQuest.

[Albermani et al., 2009] Albermani, F., Kitipornchai, S., and Chan, R. (2009). Failure analysis of transmission towers. *Engineering failure analysis*, 16(6):1922–1928.

[Baenziger et al., 1994] Baenziger, M., James, W., Wouters, B., and Li, L. (1994). Dynamic loads on transmission line structures due to galloping conductors. *Power Delivery, IEEE Transactions on*, 9(1):40–49.

[Chen et al., 2014] Chen, B., Guo, W.-H., Li, P.-Y., and Xie, W.-P. (2014). Dynamic responses and vibration control of the transmission tower-line system: a state-of-the-art review. *The Scientific World Journal*, 2014.

[Combescure and Pegon, 1997] Combescure, D. and Pegon, P. (1997). $\alpha$-operator splitting time integration technique for pseudodynamic testing error propagation analysis. *Soil Dynamics and Earthquake Engineering*, 16(7):427–443.

[Computers and Structures, 2009] Computers and Structures (2009). *CSI analysis reference manual for SAP2000, ETABS, and SAFE*. Computers and Structures, Inc.

[Cook, 1974] Cook, R. D. (1974). *Concepts and applications of finite element analysis: a treatment of the finite element method as used for the analysis of displacement, strain, and stress*. John Wiley & Sons.

[Dimig et al., 1999] Dimig, J., Shield, C., French, C., Bailey, F., and Clark, A. (1999). Effective force testing: a method of seismic simulation for structural testing. *Journal of Structural Engineering*, 125(9):1028–1037.

[Dyke et al., 1995] Dyke, S., Spencer Jr, B., Quast, P., and Sain, M. (1995). Role of control-structure interaction in protective system design. *Journal of Engineering Mechanics*, 121(2):322–338.

[Eltaly et al., 2014] Eltaly, B., Saka, A., and Kandil, K. (2014). FE simulation of transmission tower. *Advances in Civil Engineering*, 2014.

[Fan et al., 2009] Fan, Q. H. et al. (2009). Lessons learned from design and test of latticed steel transmission towers. In *Electrical Transmission and Substation Structures 2009*, pages 1–12. ASCE.

[Frankie et al., 2013] Frankie, T., Abdelnaby, A., Silva, P., Sanders, D., Elnashai, A., Spencer, B., Kuchma, D., and Chang, C. (2013). Hybrid simulation of curved four-span bridge: Comparison of numerical and hybrid experimental/analytical results and methods of numerical model calibration. In *Structures Congress*, pages 721–732.

[Gencturk and Elnashai, 2014] Gencturk, B. and Elnashai, A. (2014). Hybrid simulation of RC and ECC frames with an experimental module at small-scale. *Bridges*, 10:9780784412374–033.

[Harris and Sabnis, 2010] Harris, H. G. and Sabnis, G. (2010). *Structural modeling and experimental techniques*. CRC press.

[Hilber et al., 1977] Hilber, H. M., Hughes, T. J., and Taylor, R. L. (1977). Improved numerical dissipation for time integration algorithms in structural dynamics. *Earthquake Engineering & Structural Dynamics*, 5(3):283–292.

[Klinger et al., 2011] Klinger, C., Mehdianpour, M., Klingbeil, D., Bettge, D., Häcker, R., and Baer, W. (2011). Failure analysis on collapsed towers of overhead electrical lines in the region Münsterland (Germany) 2005. *Engineering Failure Analysis*, 18(7):1873–1883.

[Landers, 1982] Landers, P. G. (1982). EPRI-sponsored transmission line wind loading research. *Power Apparatus and Systems, IEEE Transactions on*, PAS-101(8):2460–2466.

[Logan, 2011] Logan, D. (2011). *A first course in the finite element method*. Cengage Learning.

[Mahin and Shing, 1985] Mahin, S. A. and Shing, P.-S. B. (1985). Pseudodynamic method for seismic testing. *Journal of Structural Engineering*, 111(7):1482–1503.

[Mahin et al., 1989] Mahin, S. A., Shing, P.-S. B., Thewalt, C. R., and Hanson, R. D. (1989). Pseudodynamic test method-current status and future directions. *Journal of Structural Engineering*, 115(8):2113–2128.

[Mahmoud et al., 2013] Mahmoud, H. N., Elnashai, A. S., Spencer Jr, B. F., Kwon, O.-S., and Bennier, D. J. (2013). Hybrid simulation for earthquake response of semirigid partial-strength steel frames. *Journal of Structural Engineering*, 139(7):1134–1148.

[McGuire et al., 2000] McGuire, W., Gallagher, R. H., and Ziemian, R. D. (2000). *Matrix structural analysis*. John Wiley & Sons.

[Mercado and Zhang, 2012] Mercado, M. W. and Zhang, Y. (2012). Hybrid simulation testbed for experimental validation and characterization of NDE and sensor technology. *Journal of Bridge Engineering*, 17(6):907–920.

[Mosqueda and Ahmadizadeh, 2011] Mosqueda, G. and Ahmadizadeh, M. (2011). Iterative implicit integration procedure for hybrid simulation of large nonlinear structures. *Earthquake Engineering & Structural Dynamics*, 40(9):945–960.

[PEER, 2013] PEER (2013). NGA-West2 strong-motion database. http://peer.berkeley.edu/nga_files/ath/NORTHR. Accessed April 14, 2015.

[Rao et al., 2010] Rao, N. P., Knight, G. S., Lakshmanan, N., and Iyer, N. R. (2010). Investigation of transmission line tower failures. *Engineering Failure Analysis*, 17(5):1127–1141.

[Rao et al., 2012] Rao, N. P., Knight, G. S., Mohan, S., and Lakshmanan, N. (2012). Studies on failure of transmission line towers in testing. *Engineering Structures*, 35:55–70.

[Rawlins et al., 1979] Rawlins, C., Hard, A., Ikegami, R., and Doocy, E. (1979). Transmission line reference book: wind-induced conductor motion. *Electric Power Research Institute (EPRI), Palo Alto, California.*

[Richardson, 1987] Richardson, A. (1987). Longitudinal dynamic loading of a steel pole transmission line. *Power Delivery, IEEE Transactions on*, 2(2):425–436.

[Shao and Enyart, 2014] Shao, X. and Enyart, G. (2014). Development of a versatile hybrid testing system for seismic experimentation. *Experimental Techniques*, 38(6):44–60.

[Shao and Griffith, 2013] Shao, X. and Griffith, C. (2013). An overview of hybrid simulation implementations in NEES projects. *Engineering Structures*, 56:1439–1451.

[Shao et al., 2010] Shao, X., Reinhorn, A. M., and Sivaselvan, M. V. (2010). Real-time hybrid simulation using shake tables and dynamic actuators. *Journal of Structural Engineering*, 137(7):748–760.

[Shen et al., 2010] Shen, G.-H., Cai, C., Sun, B.-N., and Lou, W.-J. (2010). Study of dynamic impacts on transmission-line systems attributable to conductor breakage using the finite-element method. *Journal of Performance of Constructed Facilities*, 25(2):130–137.

[Shing et al., 2006] Shing, P. B., Stavridis, A., Wei, Z., Stauffer, E., Wallen, R., and Jung, R.-Y. (2006). Validation of a fast hybrid test system with substructure tests. In *17th analysis and computation specialty conference.*

[Shoraka et al., 2008] Shoraka, M. B., Charlet, A. Y., Elwood, K. J., and Haukaas, T. (2008). Hybrid simulation of the gravity load collapse of reinforced concrete frames. In *Structures Congress 2008*, pages 1–17. ASCE.

[Takanashi et al., 1975] Takanashi, K., Udagawa, K., Seki, M.-U., Okada, T., and Tanaka, H. (1975). Nonlinear earthquake response analysis of structures by a computer-actuator on-line system. *Bulletin of Earthquake Resistant Structure Research Center*, 8:1–17.

[Terzic and Stojadinovic, 2013] Terzic, V. and Stojadinovic, B. (2013). Hybrid simulation of bridge response to three-dimensional earthquake excitation followed by truck load. *Journal of Structural Engineering*, 140(8).

[Udagawa and Mimura, 1991] Udagawa, K. and Mimura, H. (1991). Behavior of composite beam frame by pseudodynamic testing. *Journal of Structural Engineering*, 117(5):1317–1334.

[Wong and Miller, 2009] Wong, C. J. and Miller, M. D., editors (2009). *Guidelines for Electrical Transmission Line Structural Loading*. American Society of Civil Engineers, Reston, VA, third edition edition.

[Yang et al., 2006] Yang, T., Stojadinovic, B., and Moehle, J. (2006). Hybrid simulation evaluation of innovative steel braced framing system. In *8th US National Conference on Earthquake Engineering, San Francisco, USA*.

# APPENDIX A: HYBRID SIMULATION FUNCTION

```
function HybridSimulation(Fs,jlimit,tol,alpha,varargin)
%HYBRIDSIMULATION  Initializes a slow pseudodynamic hybrid simulation.
%   HYBRIDSIMULATION(Fs,jlimit,tol,alpha) provides user interface
%   for hybrid simulation with excitation sampling rate "Fs" (scalar>0)
%   in Hz, maximum number of iterations "jlimit" (integer > 1) per time
%   step, convergence tolerance "tol" (scalar > 0) for iterations, and
%   numerical damping "alpha" (-1/3 <= scalar <= 0).
%
%   Prompts for input files and various other options will be given.
%
%   Results of the simulation, as well as accompanying plots, will be
%   saved in the location given by the string in "folder," which
%   will be near the location where HYBRIDSIMULATION is saved.
%
%   HYBRIDSIMULATION(Fs,jlimit,tol,alpha,verification) activates
%   the verification option. The first 4 input arguments are the same
%   as previously described, and "verification" is 1. If that 5th input
%   argument is given as 1, verification is performed instead of a real
%   experiment.
%
%   In order to disable nonlinear geometry, global variable NLG may be
%   assigned as 0 prior to calling the function.

clear functions
clearvars -global -except NLG vu s1
clearvars -except Fs jlimit tol alpha varargin
global NLG NLM vu s1
pause on
%% CHECKS FOR VALID INPUTS
if length(varargin)==1
    verification=cell2mat(varargin);
    if not(verification==1)
        error('Unexpected value for "verification" input argument.')
    end
elseif or(or(length(varargin)>1,length(varargin)<0),...
            or(or(~exist('Fs','var'),~exist('jlimit','var')),...
                or(~exist('tol','var'),~exist('alpha','var'))))
    error('Unexpected number of input arguments.')
else
    verification=0;
end
if not(and(isscalar(Fs),Fs>0));          error('Invalid Fs.');     end
if not(and(mod(jlimit,1)==0,jlimit>1));  error('Invalid jlimit.'); end
if not(and(isscalar(tol),tol>0));        error('Invalid tol.');    end
if not(and(isscalar(alpha),and(alpha>=-1/3,alpha<=0)))
                                         error('Invalid alpha.');  end
clearvars varargin

mfilename('fullpath');
[pathstring,~,~]=fileparts(mfilename('fullpath'));
addpath(genpath(pathstring))
```

```
52
53 %% SETUP FUNCTIONS
54 %Structure
55 trussfile=uigetfile([pathstring,...
56     '\InputFiles\ProblemSetup\*.*'],'SPECIFY TRUSS INPUT FILE');
57 extind=strfind(trussfile,'.');
58 if strcmp(trussfile(extind:extind+3),'.txt')
59     TrussInputs(trussfile);
60 else
61     trussxlsreader(trussfile);
62 end
63
64 %Specimen and Outputs
65 global Specimen outputnode nnodes
66 if isempty(Specimen)  ; SpecimenSelect; end
67 if isempty(outputnode)
68     outputnode=NodeSelect(nnodes,'nodes for output');
69 end
70
71 %Modal Analysis
72 folder=[pathstring,'\Results\',datestr(clock,30),'\'];
73 if ~isdir(folder); mkdir(folder); end
74 resultsfile=[folder,'TestResults.txt'];
75 global UndeformedNodes nelements E A
76 [L,T]=Bar3dTransformations(UndeformedNodes);
77 M=MTotal(L); %subfunction
78 K=KTotal(L,T,zeros(nelements,1),E(Specimen)*A(Specimen)/L(Specimen));
79 [ModeShapes,NaturalFrequencies,Periods]=ModalAnalysis(K,M);
80 msgtext=strcat('The fundamental period is#',num2str(Periods(1)),...
81     '#s. If you need to prepare excitation input files, go do',...
82     '#that now. Click the button when you are ready to continue.');
83 msgtext(strfind(msgtext,'#'))=' ';
84 periodmsg=helpdlg(msgtext,'');
85 uiwait(periodmsg)
86 ModeShapePlot(ModeShapes,NaturalFrequencies,folder,[],[])
87
88 %Excitation Import
89 [Fexcite,Excitations]=ExcitationDefs(M,Fs,pathstring);
90
91 %Frequency Analysis
92 [~,nloads]=size(Excitations);
93 for i=1:nloads
94 TXYZ=Excitations(i).TXYZ;
95 ExcitationFreqAnalysis(TXYZ,NaturalFrequencies(1))
96 saveas(gcf,[folder,'\ExcitationFreqAnalysis',num2str(i),'.fig'])
97 uiwait(gcf)
98 end
99
100 %Damping
101 ready=0;
102 while ready==0
103 [ALPHA,BETA]=DampingSetup;
104 goahead=questdlg('Proceed with this damping?','','Yes','Retry','Yes');
105 if strcmp(goahead,'Yes'); ready=1; end
```

```
106 end
107 saveas(gcf,[folder,'\Damping','.fig'])
108 close(gcf)
109
110 %Re-plot Modal Analysis and Frequency Analysis for Report Purposes
111 ModeShapePlot(ModeShapes,NaturalFrequencies,folder,ALPHA,BETA)
112 loadandexcitationfreq(Excitations,folder)
113
114 %% SIMULATION AND RESULTS
115 global yieldload kpercent typunits FreeDOFs rhist dhist...
116     Elements prefunits Rho sscurve plasticlength Tult%#ok<*NUSED>
117 switch verification
118 case 0
119     %Normal operation
120     try
121     runinfo=IntegScheme(resultsfile,M,ALPHA,BETA,...
122         alpha,TXYZ(:,1),Fexcite,jlimit,tol,0);
123     saveas(sscurve,[folder,'\stressstrain.fig'])
124     close(sscurve)
125     AnalysisResults=importdata(resultsfile,'\t');
126     ResultsPlots(AnalysisResults,folder);
127     keep=questdlg('Save workspace?','','Yes','No','Yes');
128     if strcmp(keep,'Yes')
129         save([folder,'\Workspace.mat'])
130     else
131         save([folder,'\RunInfo.mat'],runinfo)
132     end
133     catch myb
134         save([folder,'\Workspace.mat'])
135         rethrow(myb)
136     end
137 case 1
138     %Verification
139     try
140     MATLAB=struct('LMLG',{},'NLMLG',{},'LMNLG',{},'NLMNLG',{});
141     RUNINFO=struct('LMLG',{},'NLMLG',{},'LMNLG',{},'NLMNLG',{});
142     for NLG=0:1
143         for NLM=0:1
144         runinfo=IntegScheme(resultsfile,M,ALPHA,BETA,...
145             alpha,TXYZ(:,1),Fexcite,jlimit,tol,1);
146         sscurvefile=strcat('\stressstrain',...
147             num2str(NLM),num2str(NLG),'.fig');
148         saveas(sscurve,[folder,sscurvefile])
149         close(sscurve)
150         TestResults.Analysis=importdata(resultsfile,'\t');
151         TestResults.RunInfo=runinfo;
152         [MATLAB,RUNINFO]=Sorter... %subfunction
153             (TestResults,NLG,NLM,MATLAB,RUNINFO);
154         end
155     end
156     try
157     [SAP,err]=SAP2000analysis(Excitations,ALPHA,BETA,alpha,jlimit,tol);
158     catch myb
159     warndlg('Error in a SAP2000 command. Plotting results anyway.')
```

```
160         for k=1:length(outputnode)
161         SAP(k).LMLG=zeros(size(MATLAB.LMLG)); %#ok<AGROW>
162         SAP(k).NLMLG=SAP(k).LMLG; %#ok<AGROW>
163         SAP(k).LMNLG=SAP(k).LMLG; %#ok<AGROW>
164         SAP(k).NLMNLG=SAP(k).LMLG; %#ok<AGROW>
165         end
166         rethrow(myb)
167     end
168     if exist('err','var')
169         if err~=0
170             error('SAP2000 command(s) failed')
171         end
172     end
173     VerificationPlots(SAP,MATLAB,folder)
174     save([folder,'\VerificationWorkspace.mat'])
175     keep=questdlg('Keep results?','','Yes','No','Yes');
176     if strcmp(keep,'No')
177         rmdir(folder,'s')
178     end
179     catch myb
180         save([folder,'\VerificationWorkspace.mat'])
181         rethrow(myb)
182     end
183 end
184 pause off
185 end
186
187 %Subfunctions
188 function M=MTotal(L)
189 global A nnodes Elements Rho FreeDOFs
190 M=zeros(3*nnodes,3*nnodes);
191 for i=1:length(L)
192     m=A(i)*L(i)*Rho(i)/2; %half the mass of the element
193     m=m*eye(3,3);
194     dofN=(Elements(i,1)*3-2:Elements(i,1)*3);
195     dofF=(Elements(i,2)*3-2:Elements(i,2)*3);
196     M(dofN,dofN)=M(dofN,dofN)+m;
197     M(dofF,dofF)=M(dofF,dofF)+m;
198 end
199 M=M(FreeDOFs,FreeDOFs);
200 end
201
202 function [MATLAB,RUNINFO]=Sorter(TestResults,NLG,NLM,MATLAB,RUNINFO)
203 if and(NLG==0,NLM==0)
204 MATLAB(1).LMLG=TestResults.Analysis;
205 RUNINFO(1).LMLG=TestResults.RunInfo;
206 elseif and(NLG==0,NLM==1)
207 MATLAB(1).NLMLG=TestResults.Analysis;
208 RUNINFO(1).NLMLG=TestResults.RunInfo;
209 elseif and(NLG==1,NLM==0)
210 MATLAB(1).LMNLG=TestResults.Analysis;
211 RUNINFO(1).LMNLG=TestResults.RunInfo;
212 elseif and(NLG==1,NLM==1)
213 MATLAB(1).NLMNLG=TestResults.Analysis;
```

```
214 RUNINFO(1).NLMNLG=TestResults.RunInfo;
215 end
216 end
```

# APPENDIX B: TWO-DIMENSIONAL VERIFICATION INPUT FILE

```
ELEMENTS
1        2
3        2
E
2.00E+11
2.00E+11
NODE BC         UX         UY         UZ
1               0          0          0
2               1          0          1
3               0          0          0
NODES
1        -4         0          0
2         0         0          0
3         0         0         -4
MASS DENSITIES
7.85E+05
7.85E+05
A
8.00E-05
9.00E-03
```

# APPENDIX C: EXCITATION INPUT FILE EXAMPLE

| s | N | N | N |
|---|---|---|---|
| 0.00 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| 0.01 | 2.36E-06 | 0.00E+00 | -4.71E-05 |
| 0.02 | 1.86E-05 | 0.00E+00 | -3.72E-04 |
| 0.03 | 6.12E-05 | 0.00E+00 | -1.22E-03 |
| 0.04 | 1.40E-04 | 0.00E+00 | -2.80E-03 |
| 0.05 | 2.62E-04 | 0.00E+00 | -5.24E-03 |
| 0.06 | 4.28E-04 | 0.00E+00 | -8.56E-03 |
| 0.07 | 6.35E-04 | 0.00E+00 | -1.27E-02 |
| 0.08 | 8.74E-04 | 0.00E+00 | -1.75E-02 |
| 0.09 | 1.13E-03 | 0.00E+00 | -2.26E-02 |
| 0.10 | 1.39E-03 | 0.00E+00 | -2.78E-02 |
| 0.11 | 1.63E-03 | 0.00E+00 | -3.25E-02 |
| 0.12 | 1.81E-03 | 0.00E+00 | -3.62E-02 |
| 0.13 | 1.92E-03 | 0.00E+00 | -3.85E-02 |
| 0.14 | 1.93E-03 | 0.00E+00 | -3.87E-02 |
| 0.15 | 1.82E-03 | 0.00E+00 | -3.64E-02 |
| 0.16 | 1.56E-03 | 0.00E+00 | -3.11E-02 |
| 0.17 | 1.13E-03 | 0.00E+00 | -2.27E-02 |
| 0.18 | 5.39E-04 | 0.00E+00 | -1.08E-02 |
| 0.19 | -2.26E-04 | 0.00E+00 | 4.52E-03 |
| 0.20 | -1.15E-03 | 0.00E+00 | 2.30E-02 |
| (cont.) | | | |
| 2.48 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| 2.49 | 0.00E+00 | 0.00E+00 | 0.00E+00 |
| 2.50 | 0.00E+00 | 0.00E+00 | 0.00E+00 |

# APPENDIX D: STIFFNESS MATRIX CODE

```
function K = KTotal( L,Transformation,fx,kE )
%KTOTAL  Constructs the elastic+geometric stiffness matrix.
%   KTOTAL( L,Transformation,fx,kE) constructs the elastic+geometric
%   stiffness matrix for a truss and removes the constrained degrees of
%   freedom to obtain the stiffness matrix for free degrees of freedom.
%   An experimental member stiffness is included via kE.
%
%   L is a column vector containing each of the member lengths to be
%   used for the matrix.
%
%   Transformation is a three-dimensional array of dimensions (6,6,n).
%   Layer "n" contains the transformation matrix T* (corresponding to
%   the notation used in Logan's "Finite Element Method" text 5th ed.,
%   Section 3.7) for the nth element.
%
%   fx is a column vector containing the axial force in each member.
%
%   kE is a scalar indicating the axial stiffness of the experimental
%   member.

global nnodes Elements nelements E A FreeDOFs Specimen
K=zeros(3*nnodes);
dofN=zeros(1,3);
dofF=zeros(1,3);
for i=1:nelements
    dofN(1:3)=(Elements(i,1)*3-2:Elements(i,1)*3);
    dofF(1:3)=(Elements(i,2)*3-2:Elements(i,2)*3);
    if isempty(Specimen)
        ke=E(i)*A(i)/L(i);
        kg=fx(i)/L(i);
    elseif not(i==Specimen)
        ke=E(i)*A(i)/L(i);
        kg=fx(i)/L(i);
    else
        kg=fx(i)/L(i);
        ke=kE-kg;
    end
%     k=ke*[ 1, 0, 0,-1, 0, 0; 0, 0, 0, 0, 0, 0; 0, 0, 0, 0, 0, 0;...
%           -1, 0, 0, 1, 0, 0; 0, 0, 0, 0, 0, 0; 0, 0, 0, 0, 0, 0]...
%      +kg*[ 1, 0, 0,-1, 0, 0; 0, 1, 0, 0,-1, 0; 0, 0, 1, 0, 0,-1;...
%           -1, 0, 0, 1, 0, 0; 0,-1, 0, 0, 1, 0; 0, 0,-1, 0, 0, 1];
    k=(ke+kg)*[ 1, 0, 0,-1, 0, 0; 0, 0, 0, 0, 0, 0; 0, 0, 0, 0, 0, 0;...
               -1, 0, 0, 1, 0, 0; 0, 0, 0, 0, 0, 0; 0, 0, 0, 0, 0, 0];
    kt=Transformation(:,:,i)'*k*Transformation(:,:,i);
    K([dofN,dofF],[dofN,dofF])=K([dofN,dofF],[dofN,dofF])+kt;
end
K=K(FreeDOFs,FreeDOFs);
end
```

# APPENDIX E: INTEGRATION SCHEME CODE

```
function runinfo=IntegScheme...
                (file,M,ALPHA,BETA,alpha,t,Fext,jlimit,tol,verf)
%INTEGSCHEME  Performs the discrete-time numerical integration for a
%hybrid simulation.
%   INTEGSCHEME(file,M,ALPHA,BETA,alpha,t,Fext,jlimit,tol,verf) performs
%   the discrete-time numerical integration for a hybrid simulation. The
%   integration scheme is iterative implicit, using numerical iterations
%   on the experimental substructure for non-initial iterations and an
%   alpha-Operator Splitting predictor-corrector step for time steps
%   that fail to converge within the allotted number of iterations.
%
%   Results for each time step are stored in the text file with name
%   given by "file." The output "runinfo" contains four columns. Column
%   1 gives the simulation time corresponding to each time step. Column
%   2 gives the index n for each time step (First time step is 1, second
%   is 2, etc.). Column 3 gives the number of iterations executed in
%   each time step.Column 4 gives the final estimated axial force in the
%   experimental member for each time step.

persistent wboption
global FreeDOFs Specimen UndeformedNodes E A kprevE L0 kE NLM k0E NLG

%% SETUP
%Establishing default options if not already set:
    if isempty(NLG); NLG=1; end; if isempty(NLM); NLM=1; end
%Integration scheme settings:
    beta=(1-alpha)^2/4; gamma=(1-2*alpha)/2;
%Loading initial conditions:
    [L0,T0]=Bar3dTransformations(UndeformedNodes); %geometry
    k0E=E(Specimen)*A(Specimen)/L0(Specimen); %specimen stiffness
    fx0=zeros(length(E),1); %member forces
%Pre-allocating space and setting initial conditions:
    nmax=length(t)-1; Deltat=t(2)-t(1); runinfo=zeros(nmax,4);
    d=zeros(length(FreeDOFs),1);    v=zeros(length(FreeDOFs),1);
    a=zeros(length(FreeDOFs),1);    r=zeros(length(FreeDOFs),1);
    Lprev=L0; fxprev=fx0; kprevE=k0E; f=Fext(:,1);
%Storing initial state vectors:
    TNodalDVA=[t(1),d',v',a']; %state vectors prior to first time step
    dlmwrite(file,TNodalDVA,'delimiter','\t','newline','pc');
%Waitbar option:
    if isempty(wboption); wboption=questdlg(['Would you like a ',...
        'progress bar?'],'','Yes please','No thanks','No thanks'); end
    if strcmp(wboption,'Yes please'); wb=1; else wb=0; end
    if wb==1
        progress=waitbar(0,'Implicit Integration');
        set(progress,'Units','pixels')
        waitopos=get(progress,'OuterPosition');
        waitpos=get(progress,'Position'); scrsz=get(0,'ScreenSize');
        set(progress,'OuterPosition',...
            [waitopos(1) scrsz(4)-waitopos(4)+...
            (waitpos(2)-waitopos(2)) waitopos(3) waitopos(4)]);
```

```
52          tic;
53      end
54
55  for n=1:nmax
56  %% BEGIN TIME STEP
57  %Determining incremental applied external force vector:
58      Deltaf=Fext(:,n+1)-f;
59  %Estimating incremental acceleration:
60      Deltaa=zeros(size(a)); %initially assume no incremental accel's
61  %Resetting incremental acceleration estimate error:
62      deltaa=zeros(size(a)); %assume the Deltaa estimate is good
63
64  for j=1:jlimit
65  %% BEGIN ITERATION
66  %Adjusting incremental acceleration estimate:
67      Deltaa=Deltaa+deltaa;
68  %Computing incremental velocity and displacement:
69      Deltav=Deltat*a+gamma*Deltat*Deltaa;
70      Deltad=Deltat*v+0.5*Deltat^2*a+beta*Deltat^2*Deltaa;
71  %Updating geometry:
72      Nodes=UpdateNodes(d+Deltad); %subfunction
73      [L,T]=Bar3dTransformations(Nodes);
74  %Updating member forces:
75      fx=NaturalAxial(fx0+NLG*(fxprev-fx0),... %subfunction
76          L0+NLG*(Lprev-L0),L);
77      fx(Specimen)=ExpSub(L(Specimen)-L0(Specimen),...
78          Deltat,t,n,j,verf);
79      Deltafx=fx-(fx0+NLG*(fxprev-fx0));
80  %Updating stiffness and damping matrices:
81      K=KTotal(L0+NLG*(L-L0),T0+NLG*(T-T0),NLG*fx,k0E); %for C
82      C=ALPHA*K+BETA*M;
83      K=KTotal(L0+NLG*(L-L0),T0+NLG*(T-T0),NLG*fx,kE);
84  %Updating restoring force vector:
85      requiv=NodalEquivalent(fx,T0+NLG*(T-T0)); %subfunction
86      if NLG==1; Deltar=requiv-r; else Deltar=K*Deltad; end
87  %Preserving P-C results in case of unconverged time step:
88      if j==2
89      Result=struct('Deltaa',{Deltaa},'Deltav',{Deltav},...
90          'Deltad',{Deltad},'Deltar',{Deltar},'Deltafx',...
91          {Deltafx},'kE',{kE},'L',{L});
92      end
93  %Computing incremental accel error from unbalanced forces:
94      Mhat=M+(1+alpha)... %pseudo-mass
95          *(gamma*Deltat*C+Deltat^2*beta*K);
96      Fhat=alpha*(Deltaf-C*Deltav-Deltar)+... %unbalanced force
97           (f+Deltaf)-M*(a+Deltaa)-C*(v+Deltav)-(r+Deltar);
98      deltaa=Mhat\Fhat; %error in Deltaa
99  %Checking for convergence once the initial P-C is done:
100     if j>1
101         norm=sqrt(sum((Deltat^2*... %modified Euclidean norm
102             beta*deltaa/max(abs(Deltad))).^2)/length(d));
103         if norm<tol
104 %Replacing preserved results with the current ones:
105             Result=struct('Deltaa',{Deltaa},'Deltav',{Deltav},...
```

```
                'Deltad',{Deltad},'Deltar',{Deltar},'Deltafx',...
                {Deltafx},'kE',{kE},'L',{L});
            break
        end
    end
end
%% END ITERATION
%Add incremental items to corresponding totals from prior step:
    a=a+Result.Deltaa; v=v+Result.Deltav; d=d+Result.Deltad;
    r=r+Result.Deltar; f=f+Deltaf; fxprev=fxprev+Result.Deltafx;
%Replacing old lengths and specimen stiffness with new ones:
    Lprev=Result.L;  if or(NLM==1,NLG==1); kprevE=Result.kE; end
%Storing current state vectors:
    TNodalDVA=[t(n+1),d',v',a']; %State vectors at end of nth time step
    dlmwrite(file,TNodalDVA,'-append','delimiter','\t','newline','pc');
%Storing status for most recent iteration:
    runinfo(n,:)=[t(n+1),n,j,fxprev(Specimen)];
%Updating waitbar:
    if wb==1; waitbarupdate(n,nmax,progress,verf); end
end
%% END TIME STEP
%Deleting waitbar
    if wb==1; delete(progress); pause(1); end
end

%Subfunctions
function Nodes=UpdateNodes(new_d)
%UpdateNodes overwrites node coordinates.
global UndeformedNodes nnodes FreeDOFs
Nodes=UndeformedNodes.';
NodeNums=Nodes(1,:);
Nodes=reshape(Nodes(2:4,:),3*nnodes,1);
for i=1:length(FreeDOFs)
    Nodes(FreeDOFs(i))=Nodes(FreeDOFs(i))+new_d(i);
end
%Restore to previous formation
Nodes=reshape(Nodes,3,nnodes);
Nodes=[NodeNums;Nodes];
Nodes=Nodes.';
end

function fx=NaturalAxial(fxprev,Lprev,L)
%This subfunction is designed for axial-only elements.
%To modify this for members subject to shear and moment,
%refer to Appendix B of the McGuire textbook.
global E A
DeltaL=zeros(size(Lprev)); fx=zeros(size(fxprev));
for i=1:length(Lprev)
    k=(E(i)*A(i)+fxprev(i))/Lprev(i);
    DeltaL(i)=L(i)-Lprev(i);
    fx(i)=fxprev(i)+k*DeltaL(i);
end
end
```

```
function requiv=NodalEquivalent(fx,T)
global nnodes Elements FreeDOFs
dofN=zeros(1,3);
dofF=zeros(1,3);
requiv=zeros(3*nnodes,1);
for n=1:length(fx)
    dofN(1:3)=(Elements(n,1)*3-2:Elements(n,1)*3);
    dofF(1:3)=(Elements(n,2)*3-2:Elements(n,2)*3);
    Fx=(fx(n)*[-1,1]*T([1,4],:,n))';
    requiv([dofN,dofF])=requiv([dofN,dofF])+Fx;
end
requiv=requiv(FreeDOFs);
end

function waitbarupdate(n,nmax,progress,verf)
persistent incupdate
if isempty(incupdate)
   if verf~=1
       incupdate=1;
   else
       incupdate=50;
   end
end
if mod(n,incupdate)==0
    duration=toc;
    timeremaining=0;
    timeremaining(1)=(duration/n)*(nmax-n);
    hourremaining=floor(timeremaining(1)/3600);
    timeremaining(1)=timeremaining(1)-hourremaining*3600;
    minsremaining=floor(timeremaining(1)/60);
    secsremaining=max(ceil(timeremaining(1)-minsremaining*60-1),0);
    hrsleft=num2str(hourremaining);
    minleft=num2str(minsremaining);
    secleft=num2str(secsremaining);
    if hourremaining<10; hrs=['0',hrsleft]; else hrs=hrsleft; end
    if minsremaining<10; mns=['0',minleft]; else mns=minleft; end
    if secsremaining<10; scs=['0',secleft]; else scs=secleft; end
    ETAtext=['Implicit Integration -- ',...
        hrs,':',mns,':',scs,' remaining'];
    waitbar(n/nmax,progress,ETAtext)
else
    waitbar(n/nmax)
end
end
```

# APPENDIX F: EXPERIMENTAL SUBSTRUCTURE CODE

```
function fx=ExpSub(TotalDeltaL,deltat,t,n,j,verf)
%EXPSUB  Determines the scalar restoring force of the experimental
%substructure.
%   EXPSUB(TotalDeltaL,deltat,t,n,j,verf) determines the scalar
%   restoring force (fx) of the experimental substructure for a
%   corresponding displacement of TotalDeltaL. In the first iteration
%   (j==1) of a time step (n), the restoring force is determined
%   directly by experiment or by experiment simulation (bilinear or
%   multilinear). In subsequent iterations (j>1), the restoring force is
%   determined indirectly, using fitted polynomials as recommended by
%   Ahmadizadeh (2007).

global dhist rhist k0E kprevE kE SSF LCF FCF prefunits
persistent dcoefs0 rcoefs dispthreshold degree method

OK=1;
if j==1
%% FIRST ITERATION
if n==1
    dhist=zeros(length(t),1);  %space for displacement history
    rhist=zeros(length(t),1);  %space for load history
    if or(or(isempty(SSF),isempty(LCF)),isempty(FCF))
        SSF=inputdlg(['What percentage of the total',...
            ' member length is the specimen: ']);
        SSF=str2double(SSF{1});
        if strcmp(prefunits,'SI')
            LCF=1/0.0254; %conversion for pref'd length to inches
            FCF=1/4.4482216152605; %conv. for pref'd force to lbs
        else
            LCF=12; %conversion for pref'd length to inches
            FCF=1; %conversion for pref'd force to lbs
        end
        psi=questdlg('Are hardware units lb and in?','','Yes',...
            'No','Yes');
        if strcmp(psi,'No')
            LCFmod=inputdlg(['Enter conversion factor to get',...
                ' from inches to hardware length units: ']);
            LCFmod=str2double(LCFmod{1});
            FCFmod=inputdlg(['Enter conversion factor to get',...
                ' from pounds to hardware force units: ']);
            FCFmod=str2double(FCFmod{1});
            if or(isempty(FCFmod),isempty(LCFmod))
                error('Units conversion unsuccessful.')
            end
        else
            LCFmod=1;
            FCFmod=1;
        end
        if or(isempty(FCFmod),isempty(LCFmod))
            error('Units conversion unsuccessful.')
        elseif or(isempty(SSF),or(SSF>100,SSF<=0))
```

```
52              error('Specified strain scale factor invalid.')
53          end
54          LCF=LCF*LCFmod;
55          FCF=FCF*FCFmod;
56          SSF=SSF/100;
57      end
58      if verf==1
59          [dnoise,rnoise,method]=hardwaresetupsim; %subfunction
60      else
61          [dnoise,rnoise]=hardwaresetup; %subfunction
62      end
63      dnoise=dnoise/(LCF*SSF);
64      rnoise=rnoise/FCF;
65      dispthreshold=max(10*sqrt((dnoise^2)/1),...
66          10*sqrt((rnoise^2)/1)/k0E);
67      if strcmp(prefunits,'SI'); lunits=' m'; else lunits=' ft'; end
68      disp(['Displacement threshold: ',num2str(dispthreshold),lunits])
69  end
70  %Enforce TotalDeltaL physically.
71  if verf==1
72      ExpSim(TotalDeltaL,n,method); %subfunction
73  else
74      Exp(TotalDeltaL,n); %subfunction
75  end
76  %Fit polynomials appropriately.
77  fitstart=max(1,(n+1)-4); %The number of data points may be modified.
78  if n==1; degree=1; else degree=2; end
79  dcoefs0=polyfit(t(fitstart:n+1),dhist(fitstart:n+1,1),degree);
80  rcoefs=polyfit(t(fitstart:n+1),rhist(fitstart:n+1,1),degree);
81  %Get result.
82  if abs(dhist(n+1,1)-dhist(n,1))>dispthreshold
83      kE=(rhist(n+1,1)-rhist(n,1))/(dhist(n+1,1)-dhist(n,1));
84  else
85      kE=kprevE;
86  end
87  %Adjust for incorrect displacement.
88  fx=rhist(n+1,1)+kE*(TotalDeltaL-dhist(n+1,1));
89
90  else
91  %% OTHER ITERATIONS
92  %Virtually enforce TotalDeltaL and estimate restoring force.
93  %Preferences: (1)Closest root to current time is preferred.
94  %             (2)Interpolation is better than extrapolation.
95  dcoefs=dcoefs0;
96  dcoefs(1,degree+1)=dcoefs(1,degree+1)-TotalDeltaL;
97  if degree==1
98      dfitroots=zeros(2,1);
99      dfitroots(1)=-dcoefs(2)/dcoefs(1);
100     dfitroots(2)=-dcoefs(2)/dcoefs(1);
101 elseif abs(dcoefs(1)/dcoefs(2))<0.0001 %(i.e. if a/b is approx. 0)
102     dfitroots=zeros(2,1);
103     dfitroots(1)=-dcoefs(3)/dcoefs(2);
104     dfitroots(2)=-dcoefs(3)/dcoefs(2);
105 else
```

```
106     dfitroots=complex(zeros(2,1));
107     dfitroots(1)=(-dcoefs(2)-sqrt(complex(dcoefs(2)^2-...
108         4*dcoefs(1)*dcoefs(3))))/(2*dcoefs(1));
109     dfitroots(2)=(-dcoefs(2)+sqrt(complex(dcoefs(2)^2-...
110         4*dcoefs(1)*dcoefs(3))))/(2*dcoefs(1));
111 end
112 rootvar=sqrt(abs(real(dfitroots)-t(n+1)).^2+...
113                     abs(imag(dfitroots)).^2);
114 rootvarmin=min(rootvar);
115 if rootvarmin<2*deltat
116     rootind=find(rootvar==rootvarmin); %ID which entry to use
117     roottime=min(dfitroots(rootind));%#ok<FNDSB> for repeated roots
118     if degree==1
119         rval=rcoefs(1)*(roottime)+rcoefs(2);
120     else
121         rval=rcoefs(1)*(roottime^2)+rcoefs(2)*roottime+rcoefs(3);
122     end
123     fx=(real(rval)/abs(real(rval)))*abs(rval);
124 elseif n>1
125     OK=0; %no solution in the time window
126 else
127     fx=k0E*TotalDeltaL; %in case actuator hasn't moved
128 end
129 if and(abs(TotalDeltaL-dhist(n,1))>dispthreshold,OK==1)
130     kE=(fx-rhist(n,1))/(TotalDeltaL-dhist(n,1));
131 else
132     kE=kprevE;
133 end
134 %% BACKUP PLAN (if nothing else worked)
135 if ~exist('fx','var')
136     fx=rhist(n+1,1)+kE*(TotalDeltaL-dhist(n+1,1));
137 end
138 end
139 if strcmp(prefunits,'SI'); kunits=' N/m'; else kunits=' lb/ft'; end
140 if verf~=1
141     disp(['Current estimated specimen stiffness = ',...
142         num2str(kE),kunits]);
143 end
144 end
145
146 %Subfunctions
147 function [dnoise,rnoise,method]=hardwaresetupsim
148 persistent methodchosen
149 if isempty(methodchosen)
150     method=questdlg('Which simulation mode?','',...
151         'Bilinear','Multilinear','Bilinear');
152     methodchosen=method;
153 end
154 method=methodchosen;
155 if strcmp(method,'Bilinear')
156     [dnoise,rnoise]=bilinearsetup;
157 elseif strcmp(method,'Multilinear')
158     [dnoise,rnoise]=multilinearsetup;
159 end
```

```
160 end
161
162 function [dnoise,rnoise]=bilinearsetup
163 global yieldload kpercent A Specimen prefunits Tult tensfail
164 if or(or(isempty(yieldload),isempty(kpercent)),...
165         or(isempty(Tult),isempty(tensfail)))
166     beep
167     if strcmp(prefunits,'SI'); stress='Pa'; else stress='psi'; end
168         yieldstress=str2double(char(inputdlg(['Input yield '...
169           ,'stress in ',stress,': '])));
170     if strcmp(prefunits,'SI')
171         yieldload=yieldstress*A(Specimen);
172     else
173         yieldload=144*yieldstress*A(Specimen);
174     end
175     kpercent=str2double(char(inputdlg...
176         ('Input percent of E for the yielded element: ')));
177     Tult=str2double(char(inputdlg(['Tensile ',...
178         'strength is how many times the yield stress?'])));
179     tensfail=questdlg(['Set tension limit at ulti',...
180         'mate strength?'],'','Yes','No','Yes');
181     if strcmp(tensfail,'Yes'); tensfail=1; else tensfail=0; end
182 if or(or(isempty(yieldload),isempty(kpercent)),...
183     or(isempty(Tult),isempty(tensfail)))
184     error('Material properties not defined!')
185 else
186     disp(['yield stress = ',num2str(yieldstress),' ',stress,''])
187     if kpercent==0; kpercent=0.01; end
188     disp(['Post-yield specimen stiffness is ',...
189         num2str(kpercent),'% of initial stiffness.'])
190 end
191 end
192 dnoise=0;
193 rnoise=0;
194 end
195
196 function [dnoise,rnoise]=multilinearsetup
197 global yieldload A Specimen prefunits plasticlength Tult
198 if or(or(isempty(yieldload),isempty(Tult)),isempty(plasticlength))
199     beep
200     if strcmp(prefunits,'SI'); stress='Pa'; else stress='psi'; end
201         yieldstress=str2double(char(inputdlg(['Input yield '...
202           ,'stress in ',stress,': '])));
203     if strcmp(prefunits,'SI')
204         yieldload=yieldstress*A(Specimen);
205     else
206         yieldload=144*yieldstress*A(Specimen);
207     end
208     plasticlength=str2double(char(inputdlg(['Plastic '...
209           ,'region length is how many times the yield strain?'])));
210     Tult=str2double(char(inputdlg(['Tensile ',...
211         'strength is how many times the yield stress?'])));
212 if or(or(isempty(yieldload),isempty(Tult)),isempty(plasticlength))
213     error('Material properties not defined!')
```

```
214 end
215 end
216 dnoise=0;
217 rnoise=0;
218 end
219
220 function [dnoise,rnoise]=hardwaresetup
221 global vu s1
222 %Tektonic Control
223 if isempty(vu)
224     vu = visa('ni','USB0::0x0699::0x0390::C010393::INSTR');
225 end
226 %Initialize the serial port
227 if isempty(s1)
228     s1 = serial('COM4','BaudRate',38400,'DataBits',8,'StopBits',1);
229 end
230 [dnoise,rnoise]=HybridMTS(0,0);
231 end
232
233 function ExpSim(d,n,method)
234 persistent speclength
235 global dhist rhist prefunits A L0 Specimen sscurve SSF LCF FCF
236 if n==1
237     if isempty(speclength)
238     speclength=SSF*L0(Specimen);
239     end
240     %-----------------------------------------------------------------
241     %For different hardware, interchange these functions:
242     if strcmp(method,'Bilinear')
243         [doutput,routput]=BilinearMTS(0,speclength);
244     elseif strcmp(method,'Multilinear')
245         [doutput,routput]=MultilinearMTS(0,speclength);
246     end
247     %-----------------------------------------------------------------
248     dhist(1,1)=doutput/(LCF*SSF);
249     rhist(1,1)=routput/FCF;
250 end
251
252 %The displacement of the hardware relative to the initial position
253 %of the actuator shall be output by the function here, as well as
254 %the absolute force measured by the hardware shall be output by the
255 %function here. Inputs and outputs here are based on an assumed
256 %hardware sign convention of + tension and - compression.
257 %-------------------------------------------------------------------
258 %For different hardware, interchange these functions:
259 if strcmp(method,'Bilinear')
260     [doutput,routput]=BilinearMTS(LCF*SSF*d,speclength);
261 elseif strcmp(method,'Multilinear')
262     [doutput,routput]=MultilinearMTS(LCF*SSF*d,speclength);
263 end
264 %-------------------------------------------------------------------
265
266 %This is for units conversion back to the user-preferred units and
267 %for scaling the displacement as needed if only part of the
```

```
%experimental member is tested.
dhist(n+1,1)=doutput/(LCF*SSF);
rhist(n+1,1)=routput/FCF;

%Plotting stress-strain:
stress=rhist(n:n+1,1)/A(Specimen);
strain=dhist(n:n+1,1)/L0(Specimen);
if n==1
    if strcmp(prefunits,'SI')
        stressunits='Pa'; strainunits='m/m';
    else
        stressunits='psi'; strainunits='in/in';
    end
    sscurve=figure;
    clf('reset')
    hold on
    title('Stress-Strain Curve')
    xlabel(strcat('\epsilon (',strainunits,')'))
    ylabel(strcat('\sigma (',stressunits,')'))
    scrsz=get(0,'ScreenSize');
    set(sscurve,'Units','pixels')
    sspos=get(sscurve,'OuterPosition');
    set(sscurve,'OuterPosition',...
        [sspos(1) scrsz(4)-103-sspos(4) sspos(3) sspos(4)]);
else
    if ~strcmp(prefunits,'SI')
        stress=stress/144;
    end
    figure(sscurve)
    hold on
end
plot(strain,stress,':xb')
end

function [D,L]=BilinearMTS(target,speclength)
global E A Specimen yieldload kpercent NLM prefunits Tult tensfail
persistent validFD
D=target;
yieldforce=yieldload;
yielddispl=(E(Specimen)*A(Specimen)/speclength)\yieldforce;
ultforce=yieldforce*Tult;
ultdispl=(kpercent*0.01*(yieldforce/yielddispl))\...
    (ultforce-yieldforce)+yielddispl;

%Convert yieldforce, yielddispl, and maxforce to lbs and inches
if strcmp(prefunits,'SI')
    yieldforce=yieldforce/4.4482216152605;
    yielddispl=yielddispl/0.0254;
    ultforce=ultforce/4.4482216152605;
    ultdispl=ultdispl/0.0254;
else
    yielddispl=yielddispl*12;
    ultdispl=ultdispl*12;
end
```

```
322 FDpoints=[-ultforce,-ultdispl;-yieldforce,-yielddispl;0,0;...
323             yieldforce,yielddispl;ultforce,ultdispl];
324 if isempty(validFD)
325     if or(length(unique(FDpoints(:,2)))~=length(FDpoints(:,2)),...
326             ~issorted(FDpoints(:,2)))
327         error(['Displacements on the force-displacement ',...
328             'curve must be monotonically increasing.'])
329     else
330         FDplot=figure;
331         plot(FDpoints(:,2),FDpoints(:,1))
332         uiwait(FDplot)
333         validFD=1;
334     end
335 end
336 if NLM==0
337     point2=find(FDpoints(:,1)==0);
338 else
339     for i=2:length(FDpoints)
340         if D<=FDpoints(i,2)
341             point2=i;
342             break
343         end
344     end
345     if ~exist('point2','var')
346         if tensfail==1
347             error('Specimen failed in tension')
348         else
349             point2=length(FDpoints(:,1));
350         end
351     end
352 end
353 kspec=(FDpoints(point2,1)-FDpoints(point2-1,1))/...
354     (FDpoints(point2,2)-FDpoints(point2-1,2));
355 L=FDpoints(point2,1)-kspec*(FDpoints(point2,2)-D);
356 end
357
358 function [D,L]=MultilinearMTS(target,speclength)
359 global E A Specimen yieldload NLM prefunits plasticlength Tult
360 persistent validFD
361 D=target;
362 yieldforce=yieldload;
363 yielddispl=(E(Specimen)*A(Specimen)/speclength)\yieldforce;
364 strharddispl=(plasticlength+1)*yielddispl;
365 strhardforce=yieldforce...
366     +0.0001*(yieldforce/yielddispl)*(strharddispl-yielddispl);
367 ultforce=yieldforce*Tult;
368 ultdispl=(ultforce-strhardforce+0.047*(yieldforce/yielddispl)...
369     *strharddispl)/(0.047*(yieldforce/yielddispl));
370
371 %Convert yieldforce, yielddispl, etc. to lbs and inches
372 if strcmp(prefunits,'SI')
373     yieldforce=yieldforce/4.4482216152605;
374     yielddispl=yielddispl/0.0254;
375     strhardforce=strhardforce/4.4482216152605;
```

```
376     strharddispl=strharddispl/0.0254;
377     ultforce=ultforce/4.4482216152605;
378     ultdispl=ultdispl/0.0254;
379 else
380     yielddispl=yielddispl*12;
381     strharddispl=strharddispl*12;
382     ultdispl=ultdispl*12;
383 end
384 FDpoints=[-yieldforce,-yielddispl;0,0;yieldforce,yielddispl;...
385     strhardforce,strharddispl;ultforce,ultdispl];
386 if isempty(validFD)
387     if or(length(unique(FDpoints(:,2)))~=length(FDpoints(:,2)),...
388             ~issorted(FDpoints(:,2)))
389         error(['Displacements on the force-displacement ',...
390             'curve must be monotonically increasing.'])
391     else
392         FDplot=figure;
393         plot(FDpoints(:,2),FDpoints(:,1))
394         uiwait(FDplot)
395         validFD=1;
396     end
397 end
398 if NLM==0
399     point2=find(FDpoints(:,1)==0);
400 else
401     for i=2:length(FDpoints)
402         if D<=FDpoints(i,2)
403             point2=i;
404             break
405         end
406     end
407     if ~exist('point2','var')
408         error('Specimen failed in tension')
409     end
410 end
411 kspec=(FDpoints(point2,1)-FDpoints(point2-1,1))/...
412     (FDpoints(point2,2)-FDpoints(point2-1,2));
413 L=FDpoints(point2,1)-kspec*(FDpoints(point2,2)-D);
414 end
415
416 function Exp(d,n)
417 global dhist rhist prefunits A L0 Specimen sscurve SSF LCF FCF
418 if n==1
419     %-----------------------------------------------------------------
420     %For different hardware, interchange this function:
421     [doutput,routput]=HybridMTS(0,2);
422     %-----------------------------------------------------------------
423     dhist(1,1)=doutput/(LCF*SSF);
424     rhist(1,1)=routput/FCF;
425 end
426
427 %The displacement of the hardware relative to the initial position
428 %of the actuator shall be output by the function here, as well as the
429 %absolute force measured by the hardware shall be output by the
```

```
%function here. Inputs and outputs here are based on an assumed
%hardware sign convention of + tension and - compression.
%--------------------------------------------------------------------
%For different hardware, interchange this function:
[doutput,routput]=HybridMTS(LCF*SSF*d,2);
%--------------------------------------------------------------------

%This is for units conversion back to the user-preferred units and
%for scaling the displacement as needed if only part of the
%experimental member is tested.
dhist(n+1,1)=doutput/(LCF*SSF);
rhist(n+1,1)=routput/FCF;

%Plotting stress-strain:
stress=rhist(n:n+1,1)/A(Specimen);
strain=dhist(n:n+1,1)/L0(Specimen);
if n==1
    if strcmp(prefunits,'SI')
        stressunits='Pa'; strainunits='m/m';
    else
        stressunits='psi'; strainunits='in/in';
    end
    sscurve=figure;
    clf('reset')
    hold on
    title('Stress-Strain Curve')
    xlabel(strcat('\epsilon (',strainunits,')'))
    ylabel(strcat('\sigma (',stressunits,')'))
    scrsz=get(0,'ScreenSize');
    set(sscurve,'Units','pixels')
    sspos=get(sscurve,'OuterPosition');
    set(sscurve,'OuterPosition',...
        [sspos(1) scrsz(4)-103-sspos(4) sspos(3) sspos(4)]);
else
    if ~strcmp(prefunits,'SI')
        stress=stress/144;
    end
    figure(sscurve)
    hold on
end
plot(strain,stress,':xb')
end
```

# APPENDIX G: SAP2000 OAPI CODE

```
function [SAP,err]=...
    SAP2000analysis(Excitations,ALPHA,BETA,alpha,jlimit,tol)
%SAP2000ANALYSIS  Performs SAP2000 simulations via API for verification
%of the original hybrid simulation software framework.
%   SAP2000analysis(Excitations,ALPHA,BETA,alpha,jlimit,tol) performs
%   SAP2000 simulations via API for verification of the original hybrid
%   simulation software framework with LMLG, LMNLG, NLMLG, and NLMNLG
%   cases, comparable to those performed using the MATLAB functions
%   HybridSimulation.m, IntegScheme.m, and ExpSub.m (April 2015). This
%   has been verified to perform satisfactorily for space trusses and
%   planar trusses with supports completely pinned and a single
%   experimental member, but these capabilities can be expanded with
%   minimal modification to this code.
%
%   "SAP" is a structure containing the results for each of the four
%   linear/nonlinear cases. "err" is a scalar to indicate whether any of
%   the API commands were unsuccessful. "err" is zero if all commands
%   were successful and nonzero if there were any failures.

global outputnode UndeformedNodes yieldload kpercent Specimen E A Rho...
    nelements nnodes Elements FreeDOFs prefunits plasticlength Tult
[~,nloads]=size(Excitations);
Times=Excitations(1).TXYZ(:,1);
mfilename('fullpath');
[pathstring,~,~]=fileparts(mfilename('fullpath'));
for load=1:nloads
TXfilename(load)={strcat(pathstring,'\InputFiles\Excitation\TX',...
    num2str(load),'.txt')};
TYfilename(load)={strcat(pathstring,'\InputFiles\Excitation\TY',...
    num2str(load),'.txt')};
TZfilename(load)={strcat(pathstring,'\InputFiles\Excitation\TZ',...
    num2str(load),'.txt')};
TXYZ=Excitations(load).TXYZ;
TXYZ(:,2:4)=Excitations(load).SF*TXYZ(:,2:4);
TX=TXYZ(:,[1,2]);
TY=TXYZ(:,[1,3]);
TZ=TXYZ(:,[1,4]);
warning('off','all')
delete(char(TXfilename(load)),...
    char(TYfilename(load)),char(TZfilename(load)))
warning('on','all')
for i=1:length(Times)
if i==1
dlmwrite(char(TXfilename(load)),TX(i,:),...
    'delimiter','\t','newline','pc');
dlmwrite(char(TYfilename(load)),TY(i,:),...
    'delimiter','\t','newline','pc');
dlmwrite(char(TZfilename(load)),TZ(i,:),...
    'delimiter','\t','newline','pc');
else
dlmwrite(char(TXfilename(load)),TX(i,:),...
```

```
    '-append','delimiter','\t','newline','pc');
dlmwrite(char(TYfilename(load)),TY(i,:),...
    '-append','delimiter','\t','newline','pc');
dlmwrite(char(TZfilename(load)),TZ(i,:),...
    '-append','delimiter','\t','newline','pc');
end
end
end
%Creating list of materials and their properties and the list of which
%material matches each element
nmaterials=1;
elmmats=zeros(size(E));
materials(nmaterials,:)=[E(1),Rho(1)];
for i=1:nelements
foundmatch=0;
    for j=1:nmaterials
        if and(E(i)==materials(j,1),Rho(i)==materials(j,2))
        elmmats(i)=j;
        foundmatch=1;
        end
    end
    if foundmatch==0
    nmaterials=nmaterials+1;
    materials(nmaterials,:)=[E(i),Rho(i)];
    elmmats(i)=nmaterials;
    end
end

%Creating a list of sections and their properties and the list of
%which section matches each element
nsections=1;
elmsects=zeros(size(A));
sections(nsections,:)=[elmmats(1),A(1)];
for i=1:nelements
foundmatch=0;
    for j=1:nsections
        if and(elmmats(i)==sections(j,1),A(i)==sections(j,2))
        elmsects(i)=j;
        foundmatch=1;
        end
    end
    if foundmatch==0
    nsections=nsections+1;
    sections(nsections,:)=[elmmats(i),A(i)];
    elmsects(i)=nsections;
    end
end

%Generating the properties to be assigned to the link element in SAP2000
[L,~]=Bar3dTransformations(UndeformedNodes);
linkmass=A(Specimen)*L(Specimen)*Rho(Specimen);
if strcmp(prefunits,'SI')
    linkweight=9.81*linkmass;
else
```

```
106     linkweight=linkmass;
107 end
108 yieldforce=yieldload;
109 yielddispl=(E(Specimen)*A(Specimen)/L(Specimen))\yieldforce;
110 if isempty(plasticlength) %Bilinear elastic material model
111 ultforce=yieldforce*Tult;
112 ultdispl=(kpercent*0.01*(yieldforce/yielddispl))\...
113     (ultforce-yieldforce)+yielddispl;
114 FDpoints=[-ultforce,-ultdispl;-yieldforce,-yielddispl;0,0;...
115             yieldforce,yielddispl;ultforce,ultdispl];
116 else %Multilinear elastic material model
117 strharddispl=(plasticlength+1)*yielddispl;
118 strhardforce=yieldforce...
119     +0.0001*(yieldforce/yielddispl)*(strharddispl-yielddispl);
120 ultforce=yieldforce*Tult;
121 ultdispl=(ultforce-strhardforce+0.047*(yieldforce/yielddispl)...
122     *strharddispl)/(0.047*(yieldforce/yielddispl));
123 FDpoints=[-yieldforce,-yielddispl;0,0;yieldforce,yielddispl;...
124     strhardforce,strharddispl;ultforce,ultdispl];
125 end
126
127 beep
128 hysteresis=questdlg('Which hysteresis model?',...
129     '','None','Kinematic','None');
130 hysteresistype=0;
131 err=0;
132 %Start SAP2000 stuff:
133 for MaterialCase=0:1
134 feature('COM_SafeArraySingleDim', 1);
135 feature('COM_PassSafeArrayByRef', 1);
136 SapObject=actxserver('Sap2000v15.SapObject');
137 err=abs(SapObject.ApplicationStart)+err;
138 if err~=0; error('SAP2000 command failed'); end
139 SapModel=SapObject.SapModel;
140 err=abs(SapModel.InitializeNewModel)+err;
141 if err~=0; error('SAP2000 command failed'); end
142
143 if MaterialCase==0
144 sapfile=[pathstring,'\Results\SapFileLM'];
145 else
146 sapfile=[pathstring,'\Results\SapFileNLM'];
147 end
148 sapfilename=[sapfile,'.sdb'];
149 err=abs(SapModel.File.NewBlank)+err;
150 if err~=0; error('SAP2000 command failed'); end
151 if strcmp(prefunits,'SI')
152 err=abs(SapModel.SetPresentUnits(10))+err; %For N_m_C
153 if err~=0; error('SAP2000 command failed'); end
154 else
155 err=abs(SapModel.SetPresentUnits(2))+err; %For lb_ft_F
156 if err~=0; error('SAP2000 command failed'); end
157 end
158
159 %Definitions for the model:
```

```
%Materials:
PropMaterial=SapModel.PropMaterial;
for i=1:nmaterials
    matname=['mat',num2str(i)];
    err=abs(PropMaterial.AddQuick('Name',1,7,1,1,1,1,1,matname))+err;
if err~=0; error('SAP2000 command failed'); end
    err=abs(PropMaterial.SetMPIsotropic...
        (matname,materials(i,1),0,1.170E-5))+err;
if err~=0; error('SAP2000 command failed'); end
    err=abs(PropMaterial.SetWeightAndMass...
        (matname,2,materials(i,2)))+err;
if err~=0; error('SAP2000 command failed'); end
end
%Frame Sections:
PropFrame=SapModel.PropFrame;
for i=1:nsections
    sectname=['sect',num2str(i)];
    matname=['mat',num2str(sections(i,1))];
    err=abs(PropFrame.SetGeneral(sectname,matname,1,1,sections(i,2),...
        1,1,1,1,1,1,1,1,1,1,1,-1,'',''))+err;
if err~=0; error('SAP2000 command failed'); end
end
%Links/Supports:
PropLink=SapModel.PropLink;
DOF=false(6,1);
Fixed=false(6,1);
NonLinear=false(6,1);
DOF(1,1)=true();
NonLinear(1,1)=true();
Stiffness=zeros(6,1);
Stiffness(1,1)=E(Specimen)*A(Specimen)/L(Specimen);
Damping=zeros(6,1);
if strcmp(hysteresis,'None')
err=abs(PropLink.SetMultiLinearElastic...
    ('link1',DOF,Fixed,NonLinear,Stiffness,Damping,0,0,'',''))+err;
if err~=0; error('SAP2000 command failed'); end
else
err=abs(PropLink.SetMultiLinearPlastic...
    ('link1',DOF,Fixed,NonLinear,Stiffness,Damping,0,0,'',''))+err;
if err~=0; error('SAP2000 command failed'); end
if strcmp(hysteresis,'Kinematic')
hysteresistype=1;
end
end
err=abs(PropLink.SetWeightAndMass...
    ('link1',linkweight,linkmass,0,0,0))+err;
if err~=0; error('SAP2000 command failed'); end
err=abs(PropLink.SetMultiLinearPoints('link1',1,length(...
    FDpoints(:,1)),FDpoints(:,1),FDpoints(:,2),hysteresistype))+err;
if err~=0; error('SAP2000 command failed'); end
err=abs(PropLink.SetSpringData('link1',1,1))+err;
if err~=0; error('SAP2000 command failed'); end

%Setting up each of the excitations:
```

```
214 FuncTH=SapModel.Func.FuncTH;
215 LoadType=cell(3*nloads,1);
216 LoadName=cell(3*nloads,1);
217 PattName=cell(3*nloads,1);
218 Fun=cell(3*nloads,1);
219 SF=zeros(3*nloads,1);
220 TF=zeros(3*nloads,1);
221 AT=zeros(3*nloads,1);
222 CSys=cell(3*nloads,1);
223 Ang=zeros(3*nloads,1);
224 for load=1:nloads
225 %Functions:
226 err=abs(FuncTH.SetFromFile_1(strcat('Excitation',num2str(load),'X'),...
227     char(TXfilename(load)),0,0,1,2,true()))+err;
228 if err~=0; error('SAP2000 command failed'); end
229 err=abs(FuncTH.SetFromFile_1(strcat('Excitation',num2str(load),'Y'),...
230     char(TYfilename(load)),0,0,1,2,true()))+err;
231 if err~=0; error('SAP2000 command failed'); end
232 err=abs(FuncTH.SetFromFile_1(strcat('Excitation',num2str(load),'Z'),...
233     char(TZfilename(load)),0,0,1,2,true()))+err;
234 if err~=0; error('SAP2000 command failed'); end
235 %LoadPatterns:
236 name=char(Excitations(load).name);
237 name=name(1:strfind(name,'.')-1);
238 err=abs(SapModel.LoadPatterns.Add...
239     (strcat('Load',num2str(load),name,'X'),8,0))+err;
240 if err~=0; error('SAP2000 command failed'); end
241 err=abs(SapModel.LoadPatterns.Add...
242     (strcat('Load',num2str(load),name,'Y'),8,0))+err;
243 if err~=0; error('SAP2000 command failed'); end
244 err=abs(SapModel.LoadPatterns.Add...
245     (strcat('Load',num2str(load),name,'Z'),8,0))+err;
246 if err~=0; error('SAP2000 command failed'); end
247 PattName(3*load-2:3*load,1)={strcat('Load',num2str(load),name,'X');
248                              strcat('Load',num2str(load),name,'Y');
249                              strcat('Load',num2str(load),name,'Z')};
250 %LoadCases:
251 type=Excitations(load).type;
252 LoadType(3*load-2:3*load,1)={type;type;type};
253 if strcmp(type,'Accel')
254     LoadName(3*load-2:3*load,1)={'U1';'U2';'U3'};
255 elseif strcmp(type,'Load')
256     LoadName(3*load-2:3*load,1)=PattName(3*load-2:3*load,1);
257 end
258 Fun(3*load-2:3*load,1)={strcat('Excitation',num2str(load),'X');...
259                         strcat('Excitation',num2str(load),'Y');...
260                         strcat('Excitation',num2str(load),'Z')};
261 SF(3*load-2:3*load,1)=ones(3,1);
262 TF(3*load-2:3*load,1)=ones(3,1);
263 AT(3*load-2:3*load,1)=zeros(3,1);
264 CSys(3*load-2:3*load,1)={'';'';''};
265 Ang(3*load-2:3*load,1)=zeros(3,1);
266 end
267 deltat=Times(2)-Times(1);
```

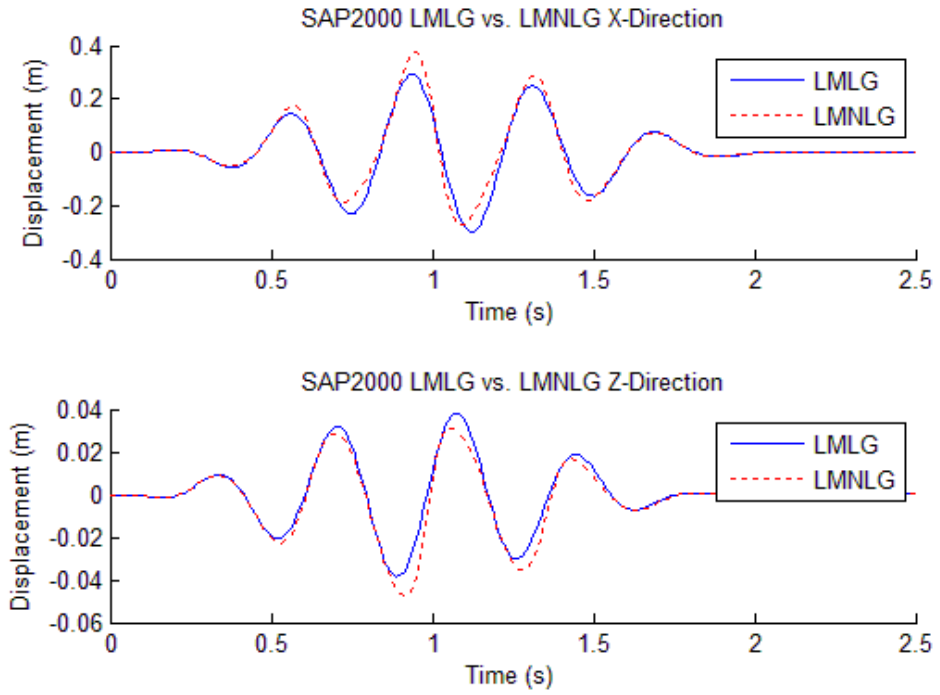
```
DirHistNonlinear=SapModel.LoadCases.DirHistNonlinear;
%linear geometry case
err=abs(DirHistNonlinear.SetCase('LG'))+err;
if err~=0; error('SAP2000 command failed'); end
err=abs(DirHistNonlinear.SetDampProportional...
    ('LG',1,BETA,ALPHA,0,0,0,0))+err;
if err~=0; error('SAP2000 command failed'); end
err=abs(DirHistNonlinear.SetGeometricNonlinearity('LG',0))+err;
if err~=0; error('SAP2000 command failed'); end
err=abs(DirHistNonlinear.SetLoads...
    ('LG',3*nloads,LoadType,LoadName,Fun,SF,TF,AT,CSys,Ang))+err;
if err~=0; error('SAP2000 command failed'); end
err=abs(DirHistNonlinear.SetTimeStep('LG',length(Times)-1,deltat))+err;
if err~=0; error('SAP2000 command failed'); end
err=abs(DirHistNonlinear.SetTimeIntegration('LG',4,alpha,0,0,1))+err;
if err~=0; error('SAP2000 command failed'); end
err=abs(DirHistNonlinear.SetSolControlParameters...
    ('LG',deltat,deltat,0,jlimit,tol,false(),1,0,1,1))+err;
if err~=0; error('SAP2000 command failed'); end
%nonlinear geometry case
err=abs(DirHistNonlinear.SetCase('NLG'))+err;
if err~=0; error('SAP2000 command failed'); end
err=abs(DirHistNonlinear.SetDampProportional...
    ('NLG',1,BETA,ALPHA,0,0,0,0))+err;
if err~=0; error('SAP2000 command failed'); end
err=abs(DirHistNonlinear.SetGeometricNonlinearity('NLG',2))+err;
if err~=0; error('SAP2000 command failed'); end
err=abs(DirHistNonlinear.SetLoads...
    ('NLG',3*nloads,LoadType,LoadName,Fun,SF,TF,AT,CSys,Ang))+err;
if err~=0; error('SAP2000 command failed'); end
err=abs(DirHistNonlinear.SetTimeStep('NLG',length(Times)-1,deltat))+err;
if err~=0; error('SAP2000 command failed'); end
err=abs(DirHistNonlinear.SetTimeIntegration('NLG',4,alpha,0,0,1))+err;
if err~=0; error('SAP2000 command failed'); end
err=abs(DirHistNonlinear.SetSolControlParameters...
    ('NLG',deltat,deltat,0,jlimit,tol,false(),1,0,1,1))+err;
if err~=0; error('SAP2000 command failed'); end

%Creating objects and making assignments:
%Nodes:
for i=1:nnodes
coords=UndeformedNodes(i,2:4);
nodename=['node',num2str(i)];
err=abs(SapModel.PointObj.AddCartesian...
    (coords(1),coords(2),coords(3),'Name',nodename,'Global'))+err;
if err~=0; error('SAP2000 command failed'); end
end
%Frames and Links:
for i=1:nelements
membername=['element',num2str(i)];
propname=['sect',num2str(elmsects(i))];
Point1=['node',num2str(Elements(i,1))];
Point2=['node',num2str(Elements(i,2))];
    if and(MaterialCase==1,i==Specimen)
```

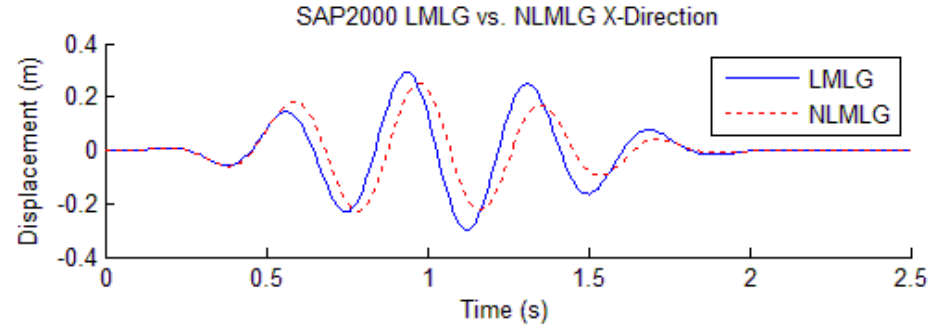
```
322     err=abs(SapModel.LinkObj.AddByPoint(Point1,Point2,'Name',...
323                                 false(),'link1',membername))+err;
324 if err~=0; error('SAP2000 command failed'); end
325     else
326     err=abs(SapModel.FrameObj.AddByPoint(Point1,Point2,'Name',...
327                                 propname,membername))+err;
328 if err~=0; error('SAP2000 command failed'); end
329     end
330 end
331 %End Releases:
332 ii=false(6,1);
333 jj=false(6,1);
334 ii(4)=true();
335 for i=5:6
336 ii(i)=true();
337 jj(i)=true();
338 end
339 startval=zeros(6,1);
340 endval=zeros(6,1);
341 err=abs(SapModel.FrameObj.SetReleases...
342     ('ALL',ii,jj,startval,endval,1))+err;
343 if err~=0; error('SAP2000 command failed'); end
344 %Restraint:
345 BCU=ones(3,nnodes);
346 for i=1:length(FreeDOFs)
347 DOF=FreeDOFs(i);
348 BCU(DOF)=0;
349 end
350 BCR=zeros(3,nnodes);
351 BC=[BCU;BCR];
352 for i=1:nnodes
353 restraints=BC(:,i);
354 val=true(6,1);
355     for j=1:6
356         if restraints(j)==0
357         val(j)=false();
358         end
359     end
360 nodename=['node',num2str(i)];
361 err=abs(SapModel.PointObj.SetRestraint(nodename,val,0))+err;
362 if err~=0; error('SAP2000 command failed'); end
363 end
364
365 %Assign Loads:
366 for load=1:nloads
367     nodenum=Excitations(load).loadnode;
368     if nodenum>0
369     err=abs(SapModel.PointObj.SetLoadForce(['node',num2str(nodenum)],...
370         char(PattName(3*load-2)),[1;0;0;0;0;0],false(),'GLOBAL',0))+err;
371 if err~=0; error('SAP2000 command failed'); end
372     err=abs(SapModel.PointObj.SetLoadForce(['node',num2str(nodenum)],...
373         char(PattName(3*load-1)),[0;1;0;0;0;0],false(),'GLOBAL',0))+err;
374 if err~=0; error('SAP2000 command failed'); end
375     err=abs(SapModel.PointObj.SetLoadForce(['node',num2str(nodenum)],...
```

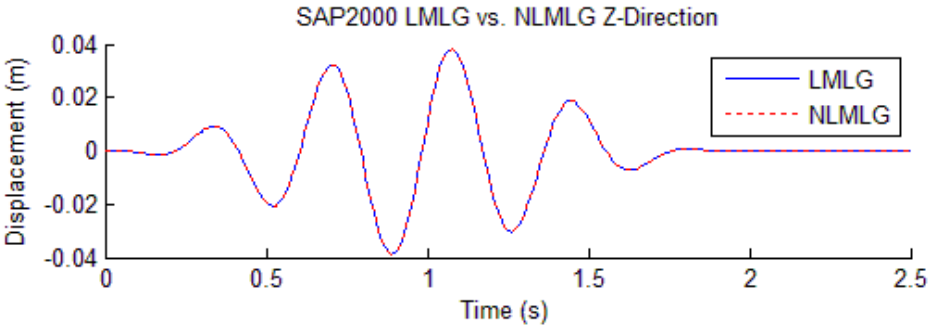
```
376         char(PattName(3*load)),[0;0;1;0;0;0],false(),'GLOBAL',0))+err;
377 if err~=0; error('SAP2000 command failed'); end
378     end
379 end
380
381 %Save:
382 err=abs(SapModel.File.Save(sapfilename))+err;
383 if err~=0; error('SAP2000 command failed'); end
384
385 %Analysis:
386 Analyze=SapModel.Analyze;
387 err=abs(Analyze.SetRunCaseFlag('Name',true(),true()))+err;
388 if err~=0; error('SAP2000 command failed'); end
389 err=abs(Analyze.RunAnalysis())+err;
390 if err~=0; error('SAP2000 command failed'); end
391
392 for k=1:length(outputnode)
393 outputjoint=['node',num2str(outputnode(k))];
394 %Results:
395 Results=SapModel.Results;
396 %linear geometry
397 err=abs(Results.Setup.DeselectAllCasesAndCombosForOutput())+err;
398 if err~=0; error('SAP2000 command failed'); end
399 err=abs(Results.Setup.SetCaseSelectedForOutput('LG',true()))+err;
400 if err~=0; error('SAP2000 command failed'); end
401 err=abs(Results.Setup.SetOptionDirectHist(2))+err;
402 if err~=0; error('SAP2000 command failed'); end
403 [~,~,~,~,~,~,StepNum,U1,U2,U3,~,~,~]=...
404     Results.JointDispl(outputjoint,0,0,{''},{''},...
405     {''},{''},0,0,0,0,0,0,0);
406 StepNum=StepNum';
407 U1=U1';
408 U2=U2';
409 U3=U3';
410 if max(abs(U2))==0
411 LGresults=[StepNum,U1,U3];
412 else
413 LGresults=[StepNum,U1,U2,U3];
414 end
415 clearvars StepNum U1 U2 U3
416 %nonlinear geometry
417 err=abs(Results.Setup.DeselectAllCasesAndCombosForOutput())+err;
418 if err~=0; error('SAP2000 command failed'); end
419 err=abs(Results.Setup.SetCaseSelectedForOutput('NLG',true()))+err;
420 if err~=0; error('SAP2000 command failed'); end
421 err=abs(Results.Setup.SetOptionDirectHist(2))+err;
422 if err~=0; error('SAP2000 command failed'); end
423 [~,~,~,~,~,~,StepNum,U1,U2,U3,~,~,~]=...
424     Results.JointDispl(outputjoint,0,0,{''},{''},...
425     {''},{''},0,0,0,0,0,0,0);
426 StepNum=StepNum';
427 U1=U1';
428 U2=U2';
429 U3=U3';
```

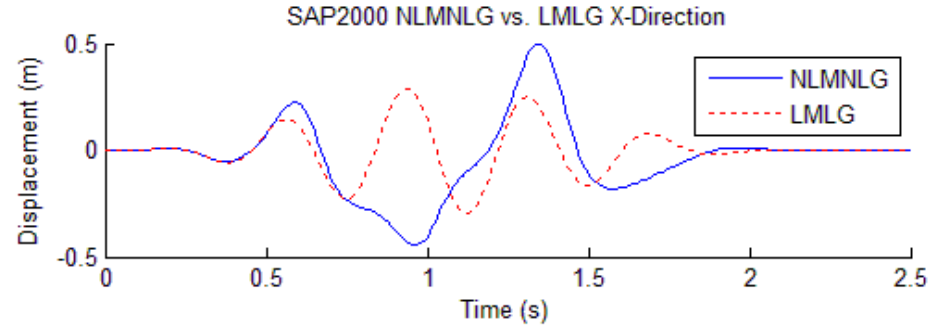
```
430 if max(abs(U2))==0
431 NLGresults=[StepNum,U1,U3];
432 else
433 NLGresults=[StepNum,U1,U2,U3];
434 end
435 clearvars StepNum U1 U2 U3
436 if MaterialCase==0
437     SAP(k).LMLG=LGresults; %#ok<*AGROW>
438     SAP(k).LMNLG=NLGresults;
439 else
440     SAP(k).NLMLG=LGresults;
441     SAP(k).NLMNLG=NLGresults;
442 end
443 end
444
445 %Delete Results:
446 err=abs(SapModel.Analyze.DeleteResults('Name',true()))+err;
447 if err~=0; error('SAP2000 command failed'); end
448
449 %Close SAP2000:
450 err=abs(SapObject.ApplicationExit(false()))+err;
451 if err~=0; error('SAP2000 command failed'); end
452 delete([sapfile,'.*'])
453 end
454 for load=1:nloads
455 delete(char(TXfilename(load)),...
456    char(TYfilename(load)),char(TZfilename(load)))
457 end
458 end
```

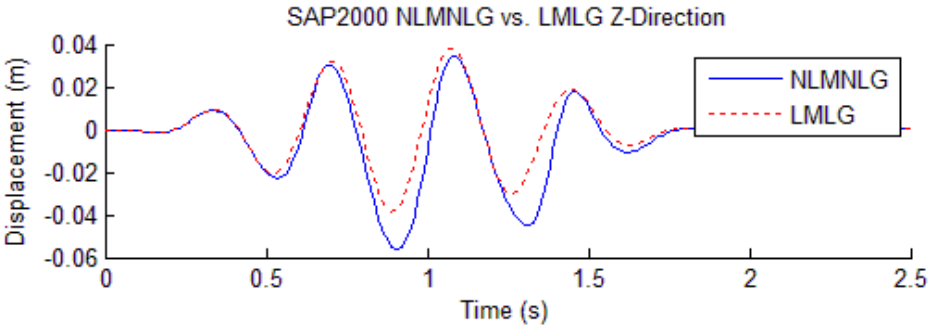
# APPENDIX H: ADDITIONAL PLOTS SUPPORTING TWO-DIMENSIONAL VERIFICATION

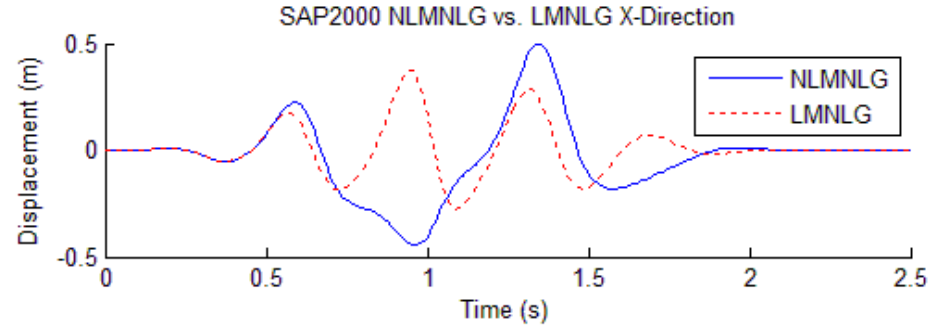


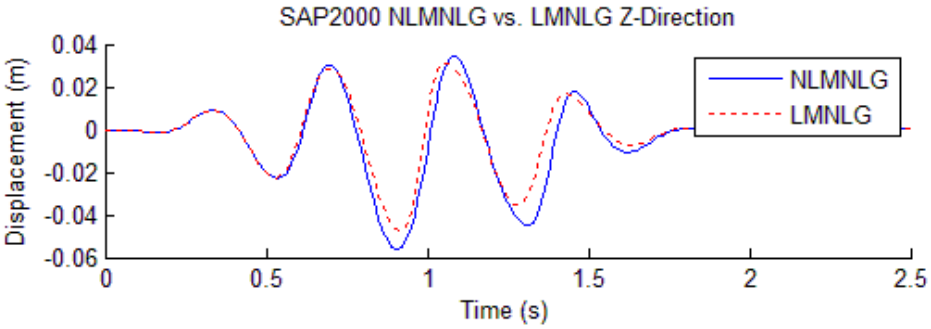

SAP2000 LMLG vs. NLMLG X-Direction
Displacement (m)
Time (s)
LMLG
NLMLG
0.4
0.2
0
-0.2
-0.4
0
0.5
1
1.5
2
2.5

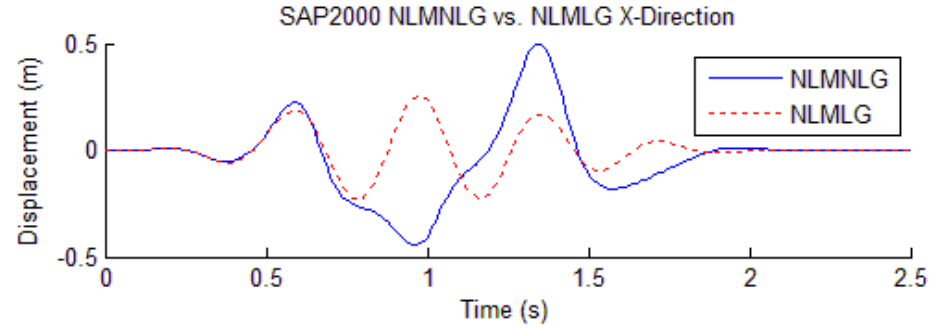

SAP2000 LMLG vs. NLMLG Z-Direction
Displacement (m)
Time (s)
LMLG
NLMLG
0.04
0.02
0
-0.02
-0.04
0
0.5
1
1.5
2
2.5

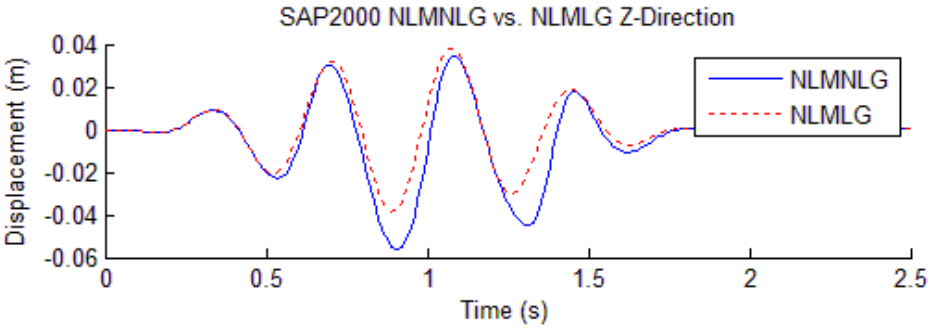

SAP2000 NLMNLG vs. LMLG X-Direction
Displacement (m)
Time (s)
NLMNLG
LMLG
0.5
0
-0.5
0
0.5
1
1.5
2
2.5

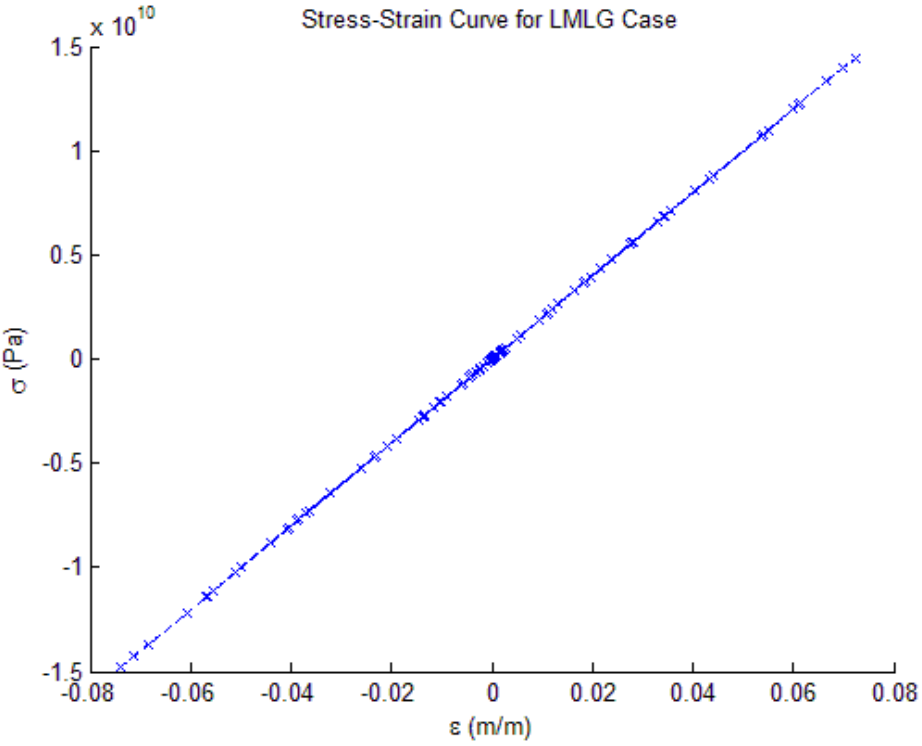

SAP2000 NLMNLG vs. LMLG Z-Direction
Displacement (m)
Time (s)
NLMNLG
LMLG
0.04
0.02
0
-0.02
-0.04
-0.06
0
0.5
1
1.5
2
2.5

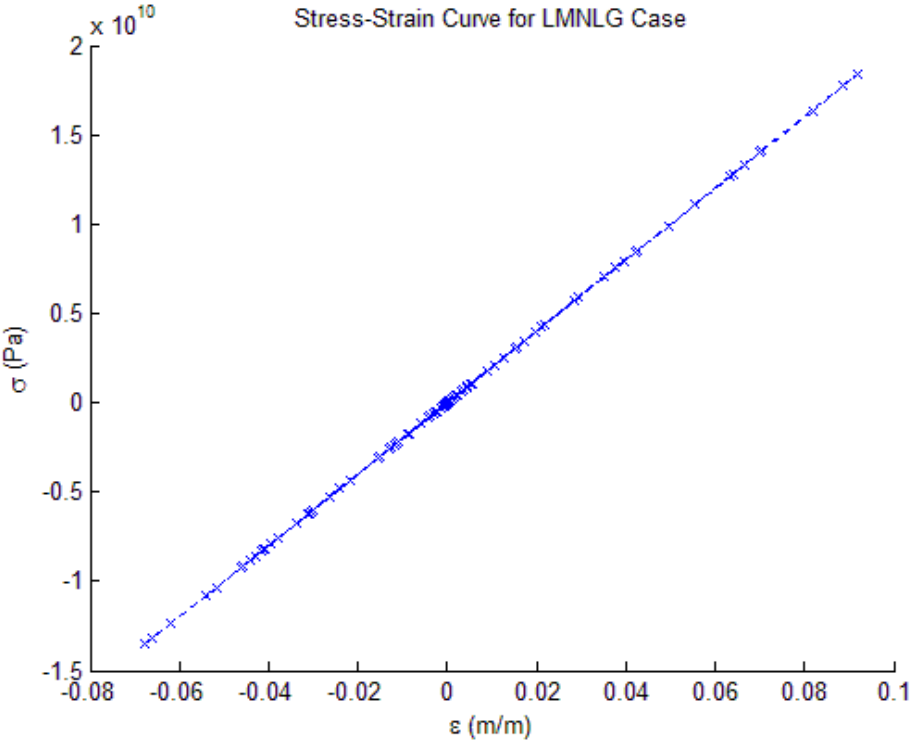

SAP2000 NLMNLG vs. LMNLG X-Direction
Displacement (m)
Time (s)
NLMNLG
LMNLG
0.5
0
-0.5
0
0.5
1
1.5
2
2.5

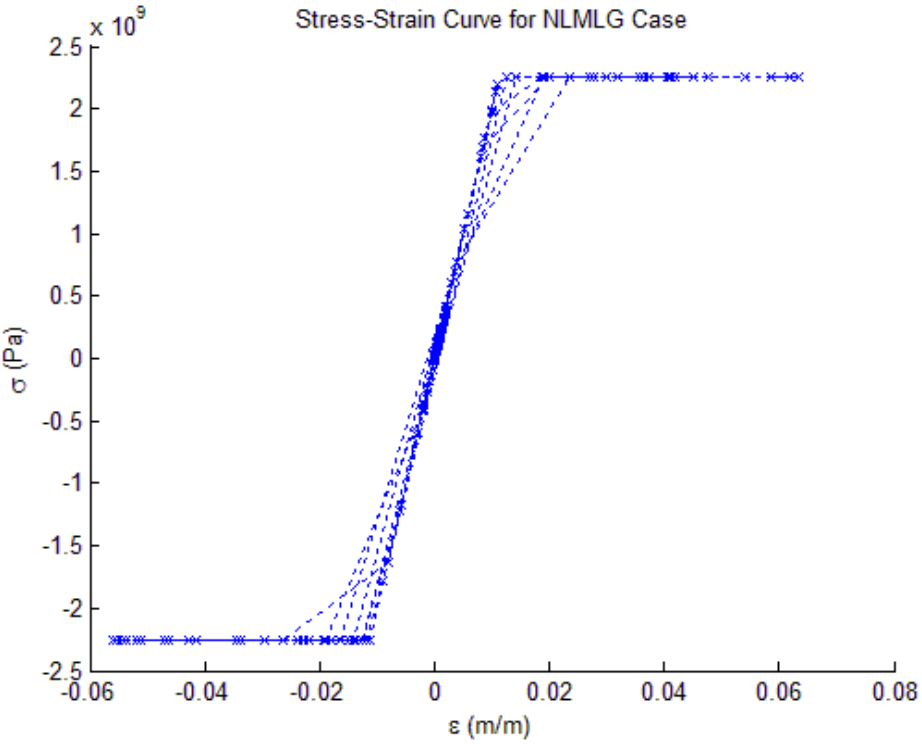

SAP2000 NLMNLG vs. LMNLG Z-Direction
Displacement (m)
Time (s)
NLMNLG
LMNLG
0.04
0.02
0
-0.02
-0.04
-0.06
0
0.5
1
1.5
2
2.5

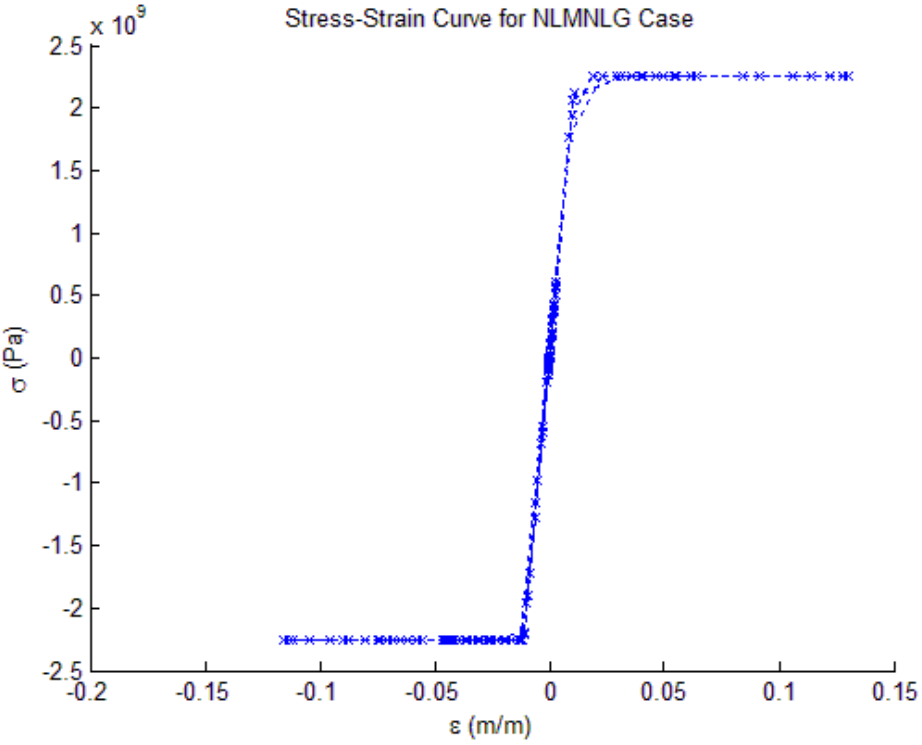

SAP2000 NLMNLG vs. NLMLG X-Direction
Displacement (m)
Time (s)
NLMNLG
NLMLG
0.5
0
-0.5
0
0.5
1
1.5
2
2.5


SAP2000 NLMNLG vs. NLMLG Z-Direction
Displacement (m)
Time (s)
NLMNLG
NLMLG
0.04
0.02
0
-0.02
-0.04
-0.06
0
0.5
1
1.5
2
2.5


Stress-Strain Curve for LMLG Case
x 10^10
1.5
1
0.5
0
-0.5
-1
-1.5
σ (Pa)
-0.08
-0.06
-0.04
-0.02
0
0.02
0.04
0.06
0.08
ε (m/m)


Stress-Strain Curve for LMNLG Case
x 10^10
2
1.5
1
0.5
0
-0.5
-1
-1.5
σ (Pa)
-0.08
-0.06
-0.04
-0.02
0
0.02
0.04
0.06
0.08
0.1
ε (m/m)


Stress-Strain Curve for NLMLG Case
x 10^9
σ (Pa)
ε (m/m)
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-0.06
-0.04
-0.02
0
0.02
0.04
0.06
0.08


Stress-Strain Curve for NLMNLG Case
x 10^9
σ (Pa)
ε (m/m)
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-0.2
-0.15
-0.1
-0.05
0
0.05
0.1
0.15

# APPENDIX I: THREE-DIMENSIONAL VERIFICATION INPUT FILE

```
ELEMENTS
1          5
2          6
3          7
4          8
5          9
6          10
7          11
8          12
9          13
10         14
11         15
12         16
13         17
14         18
15         19
16         20
17         21
18         22
19         23
20         24
1          6
2          7
3          8
4          5
5          10
6          11
7          12
8          9
9          14
10         15
11         16
12         13
13         18
14         19
15         20
16         17
17         22
18         23
19         24
20         21
5          6
6          7
```

```
7          8
8          5
9          10
10         11
11         12
12         9
13         14
14         15
15         16
16         13
17         18
18         19
19         20
20         17
21         22
22         23
23         24
24         21
5          7
9          11
13         15
17         19
21         23
E
2.00E+11
(repeated 64 times)
NODE BC         UX         UY         UZ
1               0          0          0
2               0          0          0
3               0          0          0
4               0          0          0
5               1          1          1
6               1          1          1
7               1          1          1
8               1          1          1
9               1          1          1
10              1          1          1
11              1          1          1
12              1          1          1
13              1          1          1
14              1          1          1
15              1          1          1
16              1          1          1
17              1          1          1
18              1          1          1
```

```
19              1          1          1
20              1          1          1
21              1          1          1
22              1          1          1
23              1          1          1
24              1          1          1
NODES
1          0          0          0
2          4          0          0
3          4          4          0
4          0          4          0
5          0          0          3
6          4          0          3
7          4          4          3
8          0          4          3
9          0          0          6
10         4          0          6
11         4          4          6
12         0          4          6
13         0          0          9
14         4          0          9
15         4          4          9
16         0          4          9
17         0          0          12
18         4          0          12
19         4          4          12
20         0          4          12
21         0          0          15
22         4          0          15
23         4          4          15
24         0          4          15
MASS DENSITIES
42500
(repeated 64 times)
A
9.00E-04
(repeated 19 times)
1.00E-04
(repeated 19 times)
9.00E-04
(repeated 19 times)
4.00E-04
(repeated 4 times)
```