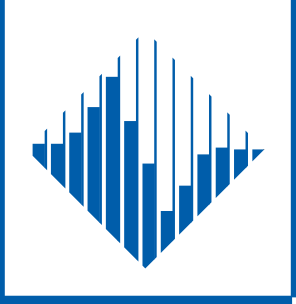


PACIFIC EARTHQUAKE ENGINEERING
RESEARCH CENTER

Advanced Implementation of Hybrid Simulation

Andreas H. Schellenberg
Stephen A. Mahin
Gregory L. Fenves
University of California, Berkeley

PEER 2009/104
NOVEMBER 2009

Advanced Implementation of Hybrid Simulation

Andreas H. Schellenberg
Department of Civil and Environmental Engineering
University of California, Berkeley

Stephen A. Mahin
Department of Civil and Environmental Engineering
University of California, Berkeley

Gregory L. Fenves
Department of Civil and Environmental Engineering
University of California, Berkeley

PEER Report 2009/104
Pacific Earthquake Engineering Research Center
College of Engineering
University of California, Berkeley

November 2009

ABSTRACT

Experimental testing of structures under simulated seismic loading is one of the best approaches to gain knowledge about the response and behavior of structures during earthquakes and to confirm the effectiveness of design methods for new earthquake-resistant structures and the retrofit of existing structures. Hybrid simulation, where a test is executed based on a step-by-step numerical solution of the governing equations of motion for a hybrid model formulated considering both the numerical and physical portions of a structural system, is one of the well-established experimental testing techniques.

In order for the earthquake engineering community to take full advantage of this technique, it is essential to conduct vigorous research to standardize the deployment of the method and extend its capabilities to applications where advanced numerical techniques are utilized; boundary conditions are imposed in real time; and dynamic loading conditions caused by wind, blast, impact, waves, fire, traffic, and, in particular, seismic events are considered. Accordingly, the objectives of this research are to investigate, develop, and validate advanced techniques to facilitate the broader use of local and geographically distributed hybrid simulation by the earthquake engineering community and to permit hybrid simulation to address a range of complex problems of current interest to researchers.

Utilizing a rigorous systems analysis of the operations performed during hybrid simulations, an existing object-oriented software framework for experimental testing (OpenFresco) is evaluated, and extensions are devised and implemented to facilitate the standardized execution of hybrid simulations. The OpenFresco middleware is environment independent, robust, flexible, and easily extensible, and supports a large variety of computational drivers, structural testing methods, specimen types, testing configurations, control and data-acquisition systems and communication protocols. One of the key contributions presented herein, is the application of a multi-tier client/server software architecture to the existing OpenFresco core components to support any finite element analysis software and facilitate an array of local and geographically distributed testing configurations.

Time-stepping integration methods that act as the computational drivers during a hybrid simulation and that need to be provided by or implemented in the finite element analysis software are thoroughly investigated. Integration schemes from both the class of direct integration methods and the class of Runge-Kutta methods are examined and their applicability

to hybrid simulation is evaluated. It is shown that the proposed operator-splitting methods and Rosenbrock methods, which are unconditionally stable, relatively easy to implement, and computationally nearly as efficient as explicit methods, are excellent techniques for solving the equations of motion during hybrid simulations.

The next part of the research focuses on the predictor-corrector algorithms that provide the synchronization of the integration and transfer system processes and that need to be implemented in a real-time environment on the control and data-acquisition system side of a hybrid simulation. Improved and new algorithms are proposed, which provide means for performing more accurate, continuous hybrid simulations, especially in situations with excessive delays. Event-driven solution strategies that employ the predictor-corrector algorithms are investigated and several new algorithms are introduced that are based on adaptive control concepts that automatically adjust parameters of the event-driven controller, such as time steps and actuator velocities, according to some suitable performance criteria.

In order to confirm the robustness, flexibility and usability of the OpenFresco software framework and to validate the integration methods and the event-driven synchronization strategies in actual tests, two novel and challenging hybrid simulation examples are carried out. The first case study performs a continuous, geographically distributed hybrid simulation in soft real-time for the first time, and the second case demonstrates how to utilize hybrid simulation to conduct safe and economical tests of structural collapse.

ACKNOWLEDGMENTS

The work in this report was made possible by financial support from a number of sources including: the Earthquake Engineering Research Centers Program of the National Science Foundation, under award number EEC-9701568 through the Pacific Earthquake Engineering Research Center (PEER); the National Science Foundation through a subaward to UC Berkeley as part of the NEES-SG: International Hybrid Simulation of Tomorrow's Braced Frame Structures (NSF 0619161); and the National Science Foundation through a subaward from the NEES Consortium Inc. (Research Agreement RA-Hybrid Sim-2007-UCB). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

1.1 MOTIVATION

The world continues to face challenges related to the rapid growth of cities and the expansion of critical infrastructure systems in seismically active regions. In order to prevent catastrophic tragedies including extensive direct (replacement and/or repair of infrastructure) and indirect (business interruption) losses due to large earthquakes, it is imperative for the civil engineering community to improve their knowledge and understanding of the response and behavior of civil structures during seismic events. Experimental testing of structures under simulated seismic loading is one of the best ways to gain such knowledge and confirm the effectiveness of design methods for new earthquake-resistant structures and the retrofit of existing ones. New models and methodologies for simulating the behavior of complex structures and advanced structural components are rapidly emerging from the fields of high-performance engineering and computing. However, it is essential to utilize results from experimental tests to validate and calibrate the new models and methodologies.

To advance the development and deployment of new and existing experimental and numerical methods in earthquake engineering, the National Science Foundation (NSF) in the United States established the George E. Brown, Jr. Network for Earthquake Engineering Simulation (NEES) as a networked collaboratory of geographically distributed, shared-use experimental research equipment sites. The resources available within this network of 15 equipment sites provide the means for collaboration and discovery in the form of advanced research based on experimentation and computational simulation of the ways buildings, bridges, utility systems, coastal regions, and geomaterials perform during seismic events [NSF].

Several of the NEES equipment sites specialize in the experimental testing and evaluation of large-scale structural and nonstructural systems utilizing hybrid simulation. In this experimental testing technique, a simulation is executed based on a step-by-step numerical solution of the governing equations of motion for a hybrid model formulated considering both the numerical and physical components of a structural system. To perform advanced hybrid

simulations of complex structural systems, the *nees@berkeley* testing facility provides a range of state-of-the-art equipment, such as a configurable reaction wall, an array of versatile static and dynamic actuators, several real-time digital controllers, powerful computational resources, and a high-speed data-acquisition system with a variety of instrumentation devices. However, in order for the earthquake engineering community to take full advantage of the versatile equipment at the *nees@berkeley* testing facility and elsewhere, extensive research must be conducted to standardize the employment of this testing method and extend hybrid simulation capabilities to applications where advanced numerical techniques are utilized; boundary conditions are imposed in real time; and dynamic loading conditions caused by wind, blast, impact, waves, fire, traffic, and seismic events are considered.

1.2 PROBLEM DEFINITION

In the past, each implementation of hybrid simulation has been problem specific and highly customized for the software and hardware available at a certain laboratory. Hence, researchers wanting to perform a hybrid simulation had to be willing to invest considerable time and effort to familiarize themselves with the available control and data-acquisition systems and to implement the problem-specific details. Many researchers have therefore been hesitant in utilizing hybrid simulation to investigate the performance of structural systems under the above-mentioned conditions. In addition, it has been challenging or often times impossible to employ existing finite element software packages to execute hybrid simulations. This is due to the lack of middleware that provides a standardized, environment-independent approach to connecting any finite element software package of a user's choice with the laboratory facilities where the experimental testing is performed. Furthermore, selecting the most appropriate time-stepping integration method, from the various for hybrid simulation specialized schemes, to solve the governing equations of motion of a certain problem type, has been very difficult as well.

To take advantage of the unique capabilities of different structural testing facilities, the geographically distributed deployment of hybrid simulation has gained significant popularity over the last few years. However, because the geographical distribution of computational and experimental sites relies heavily on wide area networks for communication, it is necessary to develop approaches for dealing with network delays and breakdowns, and for improving communication speeds. Furthermore, it is also important to standardize the geographically

distributed deployment of hybrid simulation to promote the collaboration among multi-disciplinary experts in the field of earthquake engineering, and to improve the accessibility of this testing technique to the researcher.

Because structural engineers recently are using an increasing number of rate-dependent devices to improve the performance of structures by reducing demands and/or dissipating additional energy, hybrid simulations must be performed rapidly or in real time to correctly capture the behavior of such devices. Thus, advanced methods need to be developed that are capable of imposing consistent dynamic boundary conditions with the appropriate velocities and accelerations and that are also able to deal with sporadic delays caused by the analysis software. On the other hand, if the correct execution speed cannot be achieved due to hardware or other limitations, it may be possible to develop approaches that numerically, in a truly hybrid sense, compensate for the velocity dependence of the high-performance devices.

1.3 RESEARCH OBJECTIVES AND SCOPE

The overarching goal of this research is to investigate, develop and validate advanced techniques to facilitate the wider use of local and geographically distributed hybrid simulation by the research community and to permit hybrid simulation to address a range of more complex problems of current interest to the investigators. To achieve this goal, investigations are conducted addressing the following objectives:

1. Review and evaluate existing concepts and frameworks in hybrid simulation and devise and implement an improved object-oriented software framework for experimental testing.
2. Investigate existing time-stepping integration methods, evaluate their applicability to hybrid simulations, and develop and implement improved versions for several methods.
3. Examine existing predictor-corrector algorithms and event-driven strategies for synchronization, and conceive and implement new and improved approaches to synchronize the integration and transfer system processes.

To realize these objectives, the first and main task is to devise and implement a common software framework for developing and deploying hybrid simulations. This software framework for experimental testing, which is an improved version of the one proposed by Takahashi and Fenves (2006), should support a large variety of finite element analysis software, structural testing methods, specimen types, testing configurations, control and data-acquisition systems,

and network communication protocols. Furthermore, to accelerate the development and refinement of hybrid simulation, the object-oriented software framework should be environment independent, robust, transparent, scalable, and easily extensible. Because the step-by-step numerical integration of the semi-discrete equations of motion acts as the computational driver in hybrid simulations, the second task is to investigate existing integration schemes and develop improved versions of such methods if possible. Both the direct integration methods, common to structural dynamics, and the classic Runge-Kutta methods, which are popular for solving first-order ordinary differential equations, are analyzed. The investigation of the various integration schemes should ultimately provide the means to make recommendations about selecting the most appropriate integration scheme for a certain hybrid simulation problem. To achieve continuous testing, especially in geographically distributed hybrid simulations, event-driven strategies with predictor-corrector algorithms are generally employed for synchronization. The third task is to develop improved and new predictor-corrector algorithms that are more accurate and that eliminate some of the inconsistencies of the existing methods. In addition, new event-driven strategies should be developed that can handle random time delays through adaptation. The last task is to validate the software framework, the integration methods, and the event-driven strategies by conducting a few demonstrative hybrid simulations that utilize the new developments.

1.4 ORGANIZATION OF REPORT

The advanced implementation of hybrid simulation is organized into seven chapters. Following the introduction, Chapter 2 describes the fundamental concepts of hybrid simulation, one of the current, well-established techniques for conducting laboratory tests. The chapter includes a brief summary of previous developments, the current status of hybrid simulation, a variety of advantages that are gained by using this testing technique, as well as the remaining challenges. In addition, different modes of employment, such as local versus geographically distributed and slow versus rapid testing are explained and compared, and errors affecting hybrid simulation results are summarized.

Chapter 3 presents the development details of the object-oriented framework for hybrid simulation, which is environment independent, robust, transparent, scalable, and easily extensible. The chapter presents the fundamental building blocks of the software framework and

explains the interaction among them. It also introduces the multi-tier software architecture that was employed to support a wide range of finite element analysis software packages and discusses communication methods and protocols necessary for distributed computation and testing. The chapter concludes by describing how the Open System for Earthquake Engineering Simulation (OpenSees) can be used as the computational framework in combination with the newly developed experimental framework and the advantages attained by doing so.

Specialized, time-stepping integration methods that are required to solve the semi-discrete equations of motion of a hybrid simulation model are the focus of Chapter 4. The derivation of these semi-discrete equations of motion in terms of fundamental mechanical principles is explained first. The chapter then summarizes special integration scheme properties that are necessary for the reliable, accurate, and fast execution of hybrid simulations and discusses the differences between explicit, implicit, noniterative, and iterative methods. Detailed descriptions of a number of direct integration and Runge-Kutta methods and their modifications for improving performance are provided thereafter.

Chapter 5 presents the development of improved event-driven strategies that are required to execute continuous local and geographically distributed hybrid simulations. The chapter first summarizes previous developments with an emphasis on the three-loop-architecture that was proposed for distributed testing. The theory and the analytical investigations of a variety of existing and newly developed predictor-corrector algorithms are presented thereafter. The chapter concludes by introducing server novel, adaptive, asynchronous, event-driven strategies for dealing with nondeterministic systems such as geographically distributed hybrid simulations.

Chapter 6 presents case studies of two novel applications of hybrid simulation that are made possible by the new developments described in the previous chapters. The first experimental study consists of a rapid geographically distributed hybrid simulation, which on average is executed in real time. The second study demonstrates how hybrid simulation can be utilized to perform safely and realistically experimental tests of structural collapse.

Finally, Chapter 7 presents a summary of the main findings and conclusions of this research and provides a brief list of future research directions.

2 Hybrid Simulation Fundamentals

2.1 INTRODUCTION

At present, there are several well-established methods to conduct laboratory testing for evaluating the seismic behavior of structural systems and/or components thereof. The first and most common technique is the quasi-static testing method, where the investigated structure is subjected to a predefined history of loads or displacements that are applied by actuators. By imposing the same load or displacement history on a series of specimens, the effect of systematic changes in material properties, details, boundary conditions, loading rates, and other factors can be readily identified. While such tests are relatively easy and economical to execute, the overall demands imposed on the test specimens are not directly related to the damage observed, raising questions about whether the specimens were under- or overtested for project-specific situations. Furthermore, the applied load patterns are generally inadequate for resembling the constantly changing load distributions that a structure undergoes during an actual seismic event.

Shaking table tests are a second form of laboratory tests. They are able to simulate conditions that closely resemble those that would exist during a particular earthquake. These tests provide important data on the dynamic response caused by specific ground motions, considering the inertial and energy-dissipation characteristics of the structure tested and the consequences of geometric nonlinearities, localized yielding and damage, and component failure on the structural response. However, for shaking table tests, a complete structural system is generally required, and the specimens need to be carefully constructed following the rules of dynamic similitude. On the other hand, controlling real-time dynamic component tests, which require additional actuators besides the shaking table, is particularly challenging and another topic of advanced research. So far only a few shaking tables exist with the means to apply accurate base motions to full-scale structures. The limited capacity and size of most available shaking tables place significant restrictions on the size, weight, and strength of a specimen that can be tested. As a result, reduced-scale or highly simplified specimens are commonly necessitated, which call into question the realism of many shaking table tests.

The third method is hybrid simulation, formerly also called the pseudo-dynamic test method or the online computer-controlled test method. In this experimental testing technique a simulation is executed based on a step-by-step numerical solution of the governing equations of motion for a model formulated considering both the numerical and physical components of a structural system. Contrary to conventional computational modeling and simulation, where the entire structure is analyzed analytically, the hybrid simulation method obtains the forces corresponding to either the stiffness, the energy-dissipation, or the mass properties of a structural system, from a laboratory test of the real structure or components thereof. If the hybrid simulation method is implemented for a typical quasi-static, displacement-based test, the mass and viscous damping characteristics of the specimen are numerically modeled, and the incremental displacement response of the structure to a specified dynamic excitation is computed at each step based on the current state of the physically and numerically modeled portions of the structure. During the simulation the physical portions of the overall hybrid model are tested in one or more laboratories using computer-controlled actuators, while the numerical portions are simultaneously analyzed on one or more computers. Since dynamic aspects of the simulation are handled numerically, such tests can be conducted quasi-statically using standard computer-controlled actuators. As such, hybrid simulation may be viewed as an advanced form of actuator-based testing, where the loading history is determined during the course of an experiment for a given system subjected to a specific ground motion. Alternatively, hybrid simulation can also be considered a conventional finite element analysis, where physical models of some portions of the structure are embedded in the numerical model.

A variety of advantages that are gained by utilizing hybrid simulation, instead of the quasi-static or shaking table test methods, to conduct experimental investigations are discussed in the next section. In addition, several of the challenges that still need to be addressed in hybrid simulation are summarized as well. In fact, one of the main challenges, the lack of a common framework for the development and deployment of hybrid simulation, provides the basis for the work presented in the following chapters. Afterwards, a short history on the development of the hybrid simulation testing technique since its invention three decades ago is presented. The next section then outlines the basic components and the procedure that are required to perform a hybrid simulation. Different modes of employment, such as local versus geographically distributed and slow versus rapid testing are explained and compared thereafter. Finally the

chapter concludes by presenting a brief summary of numerical and experimental errors encountered in hybrid simulations and by explaining how such errors affect a test.

2.2 ADVANTAGES AND CHALLENGES

Advantages:

Because the hybrid simulation testing technique combines analytical with experimental approaches to investigate a structural system and can be executed on expanded time-scales of up to two orders of magnitude slower than the actual time-scale, many advantages are gained. Some of the most important advantages are summarized in the following list:

- In a hybrid simulation the loading (the right-hand side of the equations of motion) is defined analytically, which provides the means to investigate the behavior of a structure excited by a wide variety of loading conditions. Seismic events including multi-support excitations; hydrodynamic loading conditions created by waves; traffic and impact loads due to moving vehicles, aerodynamic loads generated by wind and blast loads can all be simulated by incorporating them the analytical portion of the hybrid model without changing the physical portions of the experiment.
- The hybrid simulation method gives the researcher the ability to subdivide a large structure into subassemblies where the behavior is (1) well understood and can be modeled reliably using finite element models and (2) highly nonlinear and/or numerically difficult to simulate and is thus obtained from a physical test in a laboratory. This reduces modeling uncertainties by replacing numerical with actual physical components. Hence, hybrid simulation can be seen as an advanced form of component testing, which does not face the difficulties of imposing proper boundary conditions as in shaking table tests.
- Dynamic testing of full-scale specimens is made possible if the hybrid simulation method is implemented for a typical quasi-static, displacement-based test that is executed on an extended time-scale. Large-scale testing additionally eliminates the scaling difficulties encountered in shaking table tests.
- The sizes and the weights of the physical subassemblies are limited only by the available laboratory space and the strength of the strong floor and reaction frames. Furthermore, the strength of the experimental specimens is only limited by the capacities of the transfer systems (actuators) that are available at a testing facility.

- Because hybrid simulations can be executed on extended time-scales, quasi-static testing equipment including the actuators, the servo-valves, and the hydraulic power-supply are generally sufficient for testing. Oftentimes such equipment is also readily available at existing testing facilities. This and the possibility of physically testing only the critical components of a structural system make hybrid simulation a very economical technique for performing laboratory tests.
- Slow tests allow for the careful inspection of the specimens during a hybrid simulation. This can provide important insight into the behavior of a test specimen as a means to detect the initiation and track the progress of damage throughout a simulation.
- Hybrid simulations can be executed at rates anywhere between two orders of magnitude slower than the actual time scale and several times faster than the actual time scale depending on similitude requirements.
- Experimental and analytical subassemblies can be geographically distributed allowing researchers to take advantage of the different capabilities available in different laboratories.
- Effects such as geometric nonlinearities, three-dimensional effects, soil-structure interactions, and soil-pile interactions can all be incorporated into the analytical portion of the hybrid model.
- Hybrid simulation can be utilized to execute smart shaking table tests in which a simulator platform is controlled in real time to investigate problems such as tuned mass dampers or soil-structure interaction.

Challenges:

Besides the many advantages and opportunities of hybrid simulation, the method also faces several challenges that still need to be addressed by conducting further research on the technique itself and by performing specific hybrid simulations to evaluate and verify the findings. Some of these challenges are summarized in the following list:

- The development and the deployment of hybrid simulation have been hampered considerably by the absence of a common framework for conducting tests across various laboratories and computing environments. To date, each implementation of hybrid simulation is not only problem specific but also heavily dependent on the testing site and the control and data-acquisition systems used. Such highly customized software

implementations are difficult to adapt to different structural problems and even harder to port to different laboratories. Thus, there is a great need for an environment-independent software framework that is robust, transparent, and easily extensible.

- To guarantee that the results obtained from a hybrid simulation are valid, reliable and accurate, it is imperative to minimize and correct for the contamination of the solution by errors. The following errors that occur at different stages of a hybrid simulation experiment can affect the solution process: (1) modeling errors due to the discretization process, and assumptions about energy-dissipation and analysis; (2) numerical errors introduced by the integration and equilibrium solution algorithms; (3) experimental errors generated by the control and transfer systems; and (4) experimental errors introduced by the instrumentation devices and the data-acquisition system. Experimental errors can introduce energy into the hybrid system and render it unstable and must thus be minimized or compensated for at all cost.
- The execution of hybrid simulations where complex analytical subassemblies that include material and geometric nonlinearities are combined with complex inelastic experimental subassemblies is a fairly daunting task. This is because these hybrid models generally require implicit iterative integration methods, which can lead to convergence issues during the solution process. In addition, computational loads can vary significantly from time step to time step because of all the nonlinearities, making it difficult or impossible to perform rapid and real-time hybrid simulations.
- Hybrid simulations that contain very stiff or hardening physical subassemblies are difficult to execute under displacement control. Hence, it is necessary to utilize force control, mixed force/displacement control, or switching during simulation between force and displacement control. However, very little research has been conducted in these challenging areas so far.
- There is a need to develop approaches that consistently apply similitude laws if the analytical and the experimental portions of a structural system have different scales.
- Geographically distributed hybrid simulations require the means for dealing with network delays and breakdowns, and for improving communication speeds. These features are crucial for performing soft real-time geographically distributed hybrid simulations where

the real-time requirements are satisfied in an average sense but not necessarily for every single time-step.

- Another challenge is the development of methods for assessing the accuracy and the efficiency of a hybrid simulation. Such methods should allow a hybrid simulation to be evaluated not only after completion, but also during a simulation.
- As hybrid models become larger (increasing number of nodes, elements, and degrees of freedom), more complex (increasing number of material and geometric nonlinearities), and are performed near or at real time, the execution of hybrid simulations in high-performance computing (HPC) environments becomes indispensable. High-performance computing environments utilize multi-processor and/or multi-core parallel computing capabilities to distribute workloads and thus improve performance.
- In rapid and real-time hybrid simulations inertial forces are generated by the physical masses of the experimental subassemblies that should not be neglected. Thus, it is important that methods are developed to correctly account for these force contributions.
- To improve the accessibility of hybrid simulation to the experimental research community, powerful but yet comprehensible user interfaces should be developed.

2.3 HISTORY

While seeking new methods for experimentally evaluating the dynamic, especially seismic, performance of large-scale structures, several researchers initiated the development of the hybrid simulation testing technique in the early 1970s. The first official publication did not appear until 1975 when Takanashi et al. proposed the hybrid simulation testing method, then referred to as "online test," as an alternative experimental technique to shaking table tests. Since then a vast number of variations of this experimental method and the accompanying numerical algorithms have been developed to improve efficiency, accuracy, and performance. The remainder of this section presents a brief summary of such progress, which is strongly connected to the advancements achieved in developing faster and more reliable testing and computational hardware.

Because hybrid simulation test results can significantly be affected by experimental errors, the propagation of random and systematic errors were thoroughly studied (Mahin and Williams 1980; Shing and Mahin 1983) shortly after the initial developments were concluded. In

addition, Thewalt and Mahin (1987) and Mosqueda et al. (2005) provide excellent explanations and summaries on errors in hybrid simulations, including errors based on modeling, implementation techniques, and the experimental setup. Around the same time the online computer-controlled testing method also started to be referred to as the pseudo-dynamic testing technique, especially in the U.S.

In 1984, Shing and Mahin, conducted the first systematic studies in the area of numerical algorithms for the integration of the equations of motion in hybrid simulations. They investigated refinements and the implementation of stable explicit schemes, including the explicit Newmark method, which is based on the Newmark family of methods (Newmark 1959), and is still extensively used to perform hybrid simulations. It was observed that during structural testing, damage (and therefore the nonlinear behavior) often occurred in specific, limited regions of an entire structure. Hence, Dermitzakis and Mahin (1985) suggested utilizing substructuring techniques in order to divide a structure into experimental and numerical subassemblies and perform partitioned hybrid simulations. Furthermore, to overcome stability limitations imposed by explicit integration schemes applied to partitioned hybrid simulations, Dermitzakis and Mahin (1985) proposed a mixed implicit-explicit algorithm that was based on the work by Hughes and Liu (1978). Shortly thereafter Mahin and Shing (1985) published a paper that provided an excellent summary of all the U.S. activities in hybrid simulation up to that time. It presented the basic approach to the pseudo-dynamic testing method including numerical integration algorithms, implementation details, and capabilities and limitations of the technique.

After its initial development a lot of progress was made in developing hybrid simulation in both the U.S. and Japan. A whole series of papers (Nakashima 1985–1989) investigated the stability and accuracy of the pseudo-dynamic testing method. One of these publications (Takanashi and Nakashima 1987), like the paper by Mahin and Shing (1985), provided a comprehensive summary of all the Japanese activities in hybrid simulation up to that time. Significant progress was made around the same time in the U.S. by Thewalt and Mahin (1987), who executed the first multi-directional hybrid simulations, developed force and mixed force and displacement control strategies, and proposed the “effective force” dynamic testing method. However, the employment of explicit conditionally stable integration methods quickly became impractical for multi-directional hybrid simulations for multi-degrees of freedom. Thus, Thewalt and Mahin (1987) developed the first implicit unconditionally stable integration method for

hybrid simulations that was based on the alpha method by Hilber et al. (1977) and utilized analog feedback signals of the measured resisting forces instead of numerical equilibrium iterations.

While working on pseudo-dynamic tests using the substructuring technique, Nakashima also recognized the dire need for an efficient, unconditionally stable numerical algorithm to integrate the equations of motion of large multi-degrees-of-freedom hybrid models. He proposed for hybrid simulations a specialized operator-splitting (OS) method (Nakashima 1988) that was based on Hughes' work on implicit-explicit predictor-corrector schemes (Hughes et al. 1979). The stability and accuracy properties of the OS method were investigated and published shortly thereafter (Nakashima et al. 1990). It was found that the OS method possessed improved stability and error propagation properties as compared to the existing explicit methods, while not requiring any equilibrium iterations. In 1989, Mahin et al. compiled an updated summary on the current status and the future directions of hybrid simulation.

Due to the continuously increasing complexity of the structural systems that were being tested as well as the growing number of partitioned hybrid simulations, the development of implicit unconditionally stable integration schemes that were specialized for hybrid simulations was inevitable. Two researchers that developed and investigated such schemes with somewhat different approaches were Dorka and Shing. The former one proposed an integration scheme that was based on the implicit Newmark method and utilized a substepping approach in an inner loop of the algorithm (Dorka and Heiland 1991; Dorka 2002; Bayer et al. 2005). This substepping can be interpreted as a digital implementation of the force feedback, which is comparable to the analog implementation by Thewalt and Mahin (1987). The latter's integration method was based on the alpha method by Hilber et al. (1977) and employed an equilibrium solution algorithm with initial stiffness iterations (Shing et al. 1991; Shing and Vannan 1991). In addition, the equilibrium solution algorithm utilized an increment reduction factor to minimize the occurrence of spurious loading/unloading cycles. The algorithm was subsequently used to execute partitioned pseudo-dynamic tests on concentrically braced frames (Shing et al. 1994) and was extended to adaptively adjust time-step sizes during a hybrid simulation (Bursi et al. 1994).

The first implementation of hybrid simulation for real-time testing was achieved by utilizing improved hardware such as dynamic actuators and a digital servo-mechanism (Nakashima et al. 1992). In addition, these researchers employed an interlaced solution strategy, also called staggered integration, to improve computational efficiency.

In 1994, Schneider and Roeder adapted the DRAIN-2D finite element analysis software to perform hybrid simulations. This was the first time that a finite element software package was specifically modified for hybrid simulations instead of utilizing a problem-specific implementation. Furthermore, the researchers introduced the concept of replacing numerical elements with experimental elements representing the portions of the structure that are physically tested in a laboratory. Also in 1994, Thewalt and Roman, developed a technique for estimating the tangent stiffness matrix of an experimental subassembly from the measured displacements and forces. Their minimal update approach was based on the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, which belongs to the family of quasi-Newton methods, see (Bathe 1996).

Around the same time Shing et al. (1996) published a paper, which presented the hybrid simulation testing technique from a user's perspective and also provided an updated summary on the latest developments. In addition, Bursi and Shing (1996) compiled a good comparison of existing implicit integration schemes for hybrid simulation. Shing's modified implicit alpha method and Nakashima's alpha-OS method were investigated and compared in more detail. The error propagation analysis of the alpha-OS scheme, which was still missing up to this point, was performed by Combescure and Pegon (1997). To take advantage of the various equipment and the unique capabilities of different structural testing facilities, Campbell and Stojadinovic (1998) proposed geographically distributing structural subassemblies within a network of laboratories where the individual sites are connected through the internet. In the same year Magonette and Negro (1998) compiled a comprehensive summary on the latest developments and activities in hybrid simulation at the European Laboratory for Structural Assessment (ELSA).

To improve on the real-time execution of hybrid simulations, Horiuchi et al. (1996, 1999) developed actuator-delay compensation techniques that were based on actuator response predictions utilizing polynomial extrapolations. From then on real-time hybrid simulation quickly gained in popularity, which led to a constantly increasing number of researchers working on this subject. Darby et al. (1999, 2001) developed and executed several real-time partitioned hybrid simulations utilizing control system approaches. Furthermore, they proposed a real-time compatible integration algorithm for nonlinear problems that employed reduced sets of Ritz vectors in the solution process (Blakeborough et al. 2001). In addition, Nakashima and Masaoka (1999) for the first time utilized a digital signal processor (DSP) to separate the actuator signal generation task from the response analysis task. They employed an extrapolation-interpolation

procedure, based on the work by Horiuchi et al. (1996), to generate continuous displacement signals and bridge the different time scales of the two tasks. Magonette (2001), on the other hand, focused his research on developing a system to execute continuous hybrid simulations without the real-time requirement. He successfully demonstrated that continuous hybrid simulations are very effective in avoiding force relaxations and improving the accuracies of force measurements by eliminating actuator hold phases.

In 2002, Chang, proposed an unconditionally stable explicit integration scheme that was based on the Newmark family of methods. However, since the algorithm required the inverse of the mass matrix in several places, it could not be employed to solve the equations of motion of hybrid models with singular mass matrices. The three-loop architecture that was developed by Mosqueda (2003) allowed for the first time execution of continuous, geographically distributed hybrid simulations with multiple subassemblies. Compared to previous geographically distributed hybrid simulations that utilized a hold-and-ramp loading procedure, it was possible to significantly reduce execution times and eliminate force relaxation problems. In addition Mosqueda (2003) proposed several error indicators to assess the accuracy and the efficiency of a hybrid simulation. In 2004, Bonelli and Bursi, developed implicit and explicit integration schemes that were based on the generalized-alpha methods by Chung and Hulbert (1993). These integration methods provided improved adjustment of the numerical energy-dissipation characteristics of the algorithm. The first partitioned hybrid simulation with nonlinearities in both the analytical and the experimental portions of the structure were performed by Pinto et al. (2004). The research team developed a novel, parallel inter-field integration algorithm to achieve such objective.

Since up to this time most implementations of hybrid simulation had been problem specific and heavily dependent on the testing sites, several research projects were initiated to develop frameworks for conducting geographically distributed tests consisting of multiple subassemblies. Pan et al. (2005) developed an architecture where a physical test was conducted in one place, the associated numerical analysis was performed in a remote location, and the two locations communicated over the Internet. However, the proposed system adopted a file and folder sharing technique to exchange data between numerical and physical sites, sacrificing performance and flexibility. On the other hand, Takahashi and Fenves (2006) developed an object-oriented software framework consisting of a set of interrelated software classes that

provided in a convenient, modular, and standardized manner those additional functions needed by any analysis software to work with laboratory equipment to perform hybrid simulations. In addition, the software framework provided means for distributing structural subassemblies in a standardized manner within a network of laboratories and computational sites. Around the same time, Pan and coworkers improved their system by replacing the file-based data exchange routines with network socket-based ones and utilizing a centralized coordinator to enforce compatibility and equilibrium among the different subassemblies (Pan et al. 2006). This approach was similar to the one suggested earlier by Kwon et al. (2005) that consisted of a coordinator solving the equations of motion and any number of analytical and experimental static modules representing the different substructures of a hybrid model. The system provided the means to employ a wide variety of commercial finite element software codes to determine the static equilibrium of the individual modules. However, both this system as well as the one proposed by Wang et al. (2006) utilized restart capabilities provided by the finite element software packages (which write to and read from files) and therefore significantly limited performance.

Because the number of researchers and users working in hybrid simulation has increased considerably worldwide over the last few years, the number of publications has multiplied as well. Some of the research that is currently under way includes the development of force control and mixed force/displacement control strategies as well as switching between force and displacement control during simulation (Elkhoraibi and Mosalam 2007; Wu et al. 2007). Furthermore, real-time testing continues to be one of the main thrust areas in hybrid simulation (Jung et al. 2007; Mercan and Ricles 2007; Gawthrop et al. 2007; Ahmadizadeh et al. 2008; Bonnet et al. 2008; Bursi et al. 2008).

2.4 COMPONENTS AND PROCEDURE

To perform a hybrid simulation, several key components including software and hardware are necessary. These components that all interact during a hybrid simulation are shown in Fig. 2.1 and are described next.

1. The first component is a discrete model of the structure to be analyzed on a computer, including the static and dynamic loading. In the structural analysis approach presented herein, the finite element method is used to discretize the problem spatially and a time-

stepping integration algorithm is then used for the time discretization. Compared to the alternative control system approach, the finite element approach is more intuitive to the structural engineer and can easily be extended to perform partitioned hybrid simulations (see Chapter 3). The resulting dynamic equations of motion for the finite number of discrete degrees of freedom are a system of second-order time ordinary differential equations (see Chapter 4).

$$\begin{aligned} &\mathbf{M}\,\ddot{\mathbf{U}}_{i+1} + \mathbf{C}\,\dot{\mathbf{U}}_{i+1} + \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}\right) = \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} \\ &\mathbf{U}_{i=0} = \mathbf{U}_{\mathbf{0}} \\ &\dot{\mathbf{U}}_{i=0} = \dot{\mathbf{U}}_{\mathbf{0}} \end{aligned} \tag{2.1}$$

In the above equations $\mathbf{M}$ is the mass matrix assembled from the nodal and element mass matrices, $\ddot{\mathbf{U}}$ is the acceleration vector at the structural degrees of freedom, $\mathbf{C}$ is the viscous damping matrix, $\dot{\mathbf{U}}$ is the velocity vector at the structural degrees of freedom, $\mathbf{P}_{\mathbf{r}}$ are the assembled element resisting forces (which depend on the displacements), $\mathbf{P}$ are the externally applied nodal loads and $\mathbf{P}_{\mathbf{0}}$ are the assembled element loads.

2. The second required component is a transfer system consisting of a controller and static or dynamic actuators, so that the incremental response (generally the displacements) determined by the time-stepping integration algorithm can be applied to the physical portions of the structure. For slow tests, relatively inexpensive quasi-static testing equipment, which is often readily available in many structural testing laboratories, can be used for this purpose.
3. The third major component is the physical specimen that is being tested in the laboratory, including a support against which the actuators of the transfer system can react against.
4. The fourth and last component is a data-acquisition system comprising instruments such as displacement transducers, load cells, and accelerometers, to list the most common ones. The data-acquisition system is responsible for measuring the response of the test specimen and returning such experimental information (generally the resisting forces) to the time-stepping integration algorithm in order to advance the solution to the next analysis step.

Because the finite element approach is utilized to introduce the basic concepts of a hybrid simulation, the testing procedure is laid out in terms of the necessary steps to execute a direct integration analysis. To simplify the testing procedure details, a linear instead of a nonlinear

equilibrium solution algorithm was chosen in the flowchart shown in Fig. 2.2. In addition, since the Open System for Earthquake Engineering Simulation, OpenSees (McKenna 1997), finite element software is extensively used throughout this work, the corresponding commands (blue) are shown as well.

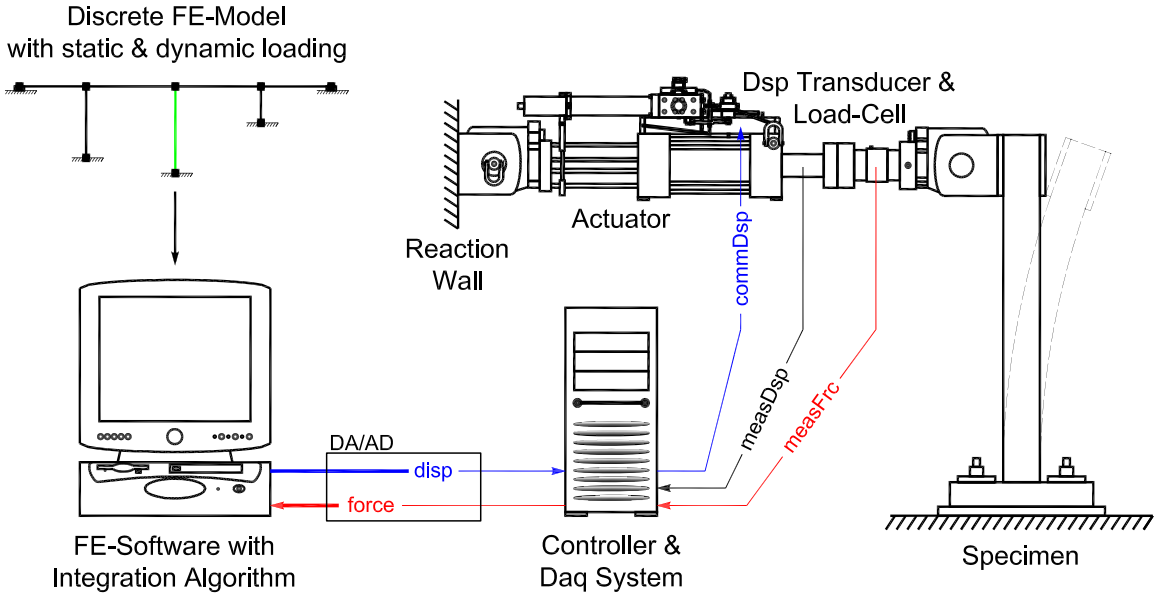


Fig. 2.1 Key components of a hybrid simulation.

As can be seen from Fig. 2.2, for each time step of the direct integration analysis, the first operation consists of determining the new trial response quantities (displacements, velocities and accelerations). Afterwards the loads and the analysis time are incremented and the new trial response quantities are sent to the analytical and experimental subassemblies. The analytical subassemblies store the new response quantities so that they can later determine the unbalanced loads. On the other hand, the experimental subassemblies communicate with the transfer systems in the laboratories, which in turn start imposing the new trial displacements.

The next task in the direct integration analysis is to solve the current time step for the response increments. Since the procedure is laid out for a linear equilibrium solution algorithm, this means that the effective tangent, mixed, or initial stiffness matrix needs to be assembled from the different portions of the structure. Depending on the integration scheme the analytical subassemblies calculate and return their tangent or initial stiffness matrices. In contrast, the experimental subassemblies return their initial stiffness matrices that have been determined numerically or analytically prior to the start of the hybrid simulation. However, if methods have been implemented for the experimental subassemblies to estimate their tangent stiffness matrices

from force and displacement measurements (Thewalt and Roman 1994), such matrices could be returned to the equilibrium solution algorithm instead.

Fig. 2.2 Hybrid simulation testing procedure.

Next, the equilibrium solution algorithm needs to assemble the unbalanced force vector. As a result, the analytical subassemblies retrieve the stored response quantities to calculate and return their resting forces and other contributions to the unbalanced force vector. On the other hand, the experimental subassemblies communicate with the data-acquisition systems in the laboratories to measure the resisting forces as soon as the previously received target response is reached. The measured resisting forces combined with the other contributions to the unbalanced force vector are then returned to the finite element analysis. Once the assembly processes of the effective stiffness matrix and the unbalanced force vector are terminated, the equilibrium solution algorithm can solve the linear system of equations for the response increments and

subsequently update the response of the entire structure. Finally, the solution state of the structure is committed and the direct integration analysis can be advanced to the next time step.

2.5 LOCAL VS. GEOGRAPHICALLY DISTRIBUTED TESTING

As has been described earlier, when all the advantages were laid out, hybrid simulation is a very versatile testing technique that facilitates many different modes of employment. Hence, it is useful to categorize some of these main modes of employment and to define the associated terminologies.

The first way of categorizing hybrid simulations is by grouping them according to the geographical distribution of the structural subassemblies and the different components. In a local hybrid simulation the numerical analysis and the experimental testing are performed at the same location, i.e., in the same laboratory. Because the finite element analysis hardware, the control system, and the data-acquisition system are all collocated, it is possible to connect them through local high-speed network rings, e.g., the Shared Common Random Access Memory Network (SCRAMNet+) (Systran). Such local high-speed connections drastically reduce delays and make it possible to perform continuous and/or real-time hybrid simulations. Local hybrid simulations are thus not defined by the absence of a network but rather by the collocation of all the required components in one laboratory.

Conversely, in a geographically distributed hybrid simulation various parts of a structure are analyzed and tested at different sites. This means that the analytical portions of the model can be analyzed on multiple computers, while the physical portions of the hybrid model are tested in multiple laboratories. Because some or all the participating sites are dispersed, they need to be connected through wide area networks such as the Internet or the Internet2's Abilene network. Generally these networks create much larger delays than the local high-speed networks and the execution of rapid hybrid simulations becomes very challenging. In addition, it is necessary to establish the means for dealing with excessive network delays as well as breakdowns. However, by breaking a model up into selected subassemblies and distributing them within a network of laboratories and computational sites, a researcher is able to take advantage of and combine the different capabilities available at the various facilities.

2.6 SLOW VS. RAPID VS. REAL-TIME TESTING

Another important feature that can be utilized to categorize a hybrid simulation is how fast a test is executed. Slow tests can be executed on extended time-scales of up to two orders of magnitude slower than the actual time-scale, whereas real-time tests need to be executed equal to or faster than the actual time-scale depending on similitude requirements. Thus, different aspects of a hybrid simulation need to be considered and treated more carefully, conditional on the execution speed. For slow tests it is imperative that the physically tested portions of the structure not exhibit any velocity-dependent behavior unless such dependency is compensated for in the analytical part of the hybrid model. In addition, it is important that an uninterrupted execution, meaning a continuous actuator motion, is maintained to avoid force relaxation and actuator stick-slip difficulties. On the other hand, for rapid hybrid simulations it is crucial that the inertial and damping force contributions generated by the physical portions of the hybrid model are correctly accounted or compensated for. Moreover, it is important to recognize that for such tests high-force dynamic actuators with large accumulators and hydraulic pumping systems are required and that the accurate control of these actuators becomes much more difficult due to inertial force feedbacks.

The equations of motion that need to be solved during a hybrid simulation take on slightly different forms depending on the execution speed of the test. For slow tests, where no inertial forces are generated in the physical subassemblies, they can be expressed in the following manner.

$$\mathbf{M}\,\ddot{\mathbf{U}}_{i+1} + \mathbf{C}\,\dot{\mathbf{U}}_{i+1} + \mathbf{P}_{\mathbf{r}}^{A}\left(\mathbf{U}_{i+1}, \dot{\mathbf{U}}_{i+1}\right) + \mathbf{P}_{\mathbf{r}}^{E}\left(\mathbf{U}_{i+1}\right) = \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} \tag{2.2}$$

where the terms with superscripts A are assembled from the analytical portions of the structure and the terms with superscripts E are assembled from the experimental portions of the structure. As can be seen from Equation (2.2) the mass and the viscous damping of the entire structure are treated numerically because the experimental resisting force vector $\mathbf{P}_{\mathbf{r}}^{E}$ does not include any inertial or viscous damping force contributions due to the slow execution of the test. Hence, the mass matrix $\mathbf{M}$ and viscous damping matrix $\mathbf{C}$ can directly be assembled from all the node and element contributions of both the analytical and experimental portions of a structure.

In contrast, if hybrid simulations are executed more rapidly, where inertial and viscous damping forces start being generated in the physically tested portions of a structure, it is

necessary to compensate for these dynamic forces, and the experimental resisting force vector $\mathbf{P}_{\mathbf{r}}^{E}$ in Equation (2.2) needs to be modified to correctly account for such effects.

$$\mathbf{P}_{\mathbf{r}}^{E}\left(\mathbf{U}_{i+1}\right)=\mathbf{P}_{\mathbf{r},i+1}^{E}-\mathbf{M}^{E}\,\ddot{\mathbf{U}}_{i+1}^{E}-\mathbf{C}^{E}\,\dot{\mathbf{U}}_{i+1}^{E} \tag{2.3}$$

In the above equation $\mathbf{P}_{\mathbf{r},i+1}^{E}$ are the resisting forces measured by the load cells and assembled from the experimental subassemblies (which include the dynamic force contributions), $\mathbf{M}^{E}$ and $\mathbf{C}^{E}$ are the mass and viscous damping matrices assembled from the experimental subassemblies, and $\ddot{\mathbf{U}}_{i+1}^{E}$ and $\dot{\mathbf{U}}_{i+1}^{E}$ are the accelerations and velocities measured from the experimental subassemblies. Even though the dynamic force correction in (2.3) is shown for the global degrees of freedom, it should ideally be performed for the element or even the actuator degrees of freedom instead.

Once the execution speed reaches real time, meaning that the experimental subassemblies are loaded with the actual, calculated velocities and accelerations (including similitude requirements), an alternative, preferred form of the equations of motion can be employed.

$$\mathbf{M}^{A}\,\ddot{\mathbf{U}}_{i+1}+\mathbf{C}^{A}\,\dot{\mathbf{U}}_{i+1}+\mathbf{P}_{\mathbf{r}}^{A}\left(\mathbf{U}_{i+1},\dot{\mathbf{U}}_{i+1}\right)+\mathbf{P}_{\mathbf{r}}^{E}\left(\mathbf{U}_{i+1},\dot{\mathbf{U}}_{i+1},\ddot{\mathbf{U}}_{i+1}\right)=\mathbf{P}_{i+1}-\mathbf{P}_{\mathbf{0},i+1} \tag{2.4}$$

where $\mathbf{M}^{A}$ and $\mathbf{C}^{A}$ are the mass and viscous damping matrices assembled from only the analytical subassemblies and $\mathbf{P}_{\mathbf{r}}^{E}$ is the resisting force vector assembled from the experimental subassemblies, which includes the inertial and damping forces of these physical portions of the structure. It is important to notice that the experimental resisting force vector is assembled entirely from measured force values but for illustrative purposes can be expressed in the following manner.

$$\mathbf{P}_{\mathbf{r}}^{E}\left(\mathbf{U}_{i+1},\dot{\mathbf{U}}_{i+1},\ddot{\mathbf{U}}_{i+1}\right)=\mathbf{P}_{\mathbf{r},i+1}^{E}+\mathbf{M}^{E}\,\ddot{\mathbf{U}}_{i+1} \tag{2.5}$$

On the other hand, if a real-time hybrid simulation is utilized to execute smart shaking table tests, where entire subassemblies are tested on simulator platforms, the experimental resisting force vector $\mathbf{P}_{\mathbf{r}}^{E}$ is no longer a function of the relative response quantities. Instead, absolute response quantities need to be sent to the control systems driving the shaking tables. Again for illustrative purposes only, this can be expressed as follows.

$$\mathbf{P}_{\mathbf{r}}^{E}\left(\mathbf{U}_{\mathbf{t},i+1},\dot{\mathbf{U}}_{\mathbf{t},i+1},\ddot{\mathbf{U}}_{\mathbf{t},i+1}\right)=\mathbf{P}_{\mathbf{r},i+1}^{E}+\mathbf{M}^{E}\,\ddot{\mathbf{U}}_{\mathbf{t},i+1} \tag{2.6}$$

For convenience the different forms of the equations of motion that need to be solved for the various types of hybrid simulations are summarized in Table 2.1 below.

Table 2.1 Equations of motion for different types of hybrid simulations.

Type of Hybrid Simulation	*Equations of Motion*
slow test	2-2
rapid test	2-2 & 2-3
real-time test	2-2 & 2-3 or 2-4 & 2-5
smart shaking table test	2-4 & 2-6

Finally, many of the numerical time-stepping integration algorithms that can be employed to solve the presented equations of motion are discussed in detail in Chapter 4.

2.7 NUMERICAL AND EXPERIMENTAL ERRORS

To guarantee that the results obtained from a hybrid simulation are valid, reliable, and accurate, it is imperative to minimize and correct for the contamination of the results by errors. In hybrid simulations errors can be introduced into the solution process at various stages and it is possible to categorize them as numerical versus experimental errors, or, alternatively, as random versus systematic errors. While numerical and random errors do not significantly affect the response of a structure and can generally be ignored, experimental and systematic errors can accumulate over the course of a hybrid simulation and lead to poor results or even instabilities. As mentioned in Section 2.3, many researchers have thoroughly investigated the effects of various errors on the structural response. Shing and Mahin (1983, 1984, 1990) concentrated their studies on numerical errors that are introduced by the model abstraction and by the time-stepping integration algorithms. Experimental errors introduced by hardware components of the transfer and data-acquisition systems were investigated and summarized by Thewalt and Mahin (1987). Mosqueda (2003) performed extensive numerical simulations of experimental errors to investigate their effects on the results of a hybrid simulation. He derived analytical models in the form of linear transfer functions for time delay errors and for the dynamic behavior of experimental setups (including the specimen, the transfer system and the reaction wall) to investigate their effect on the overall performance of the transfer system.

Because comprehensive summaries of hybrid simulation errors and the available error compensation techniques have been compiled by Mosqueda (2003), only a brief list of numerical and experimental errors is presented here for convenience.

- The first source of errors is the spatial discretization of the structure that is necessary to obtain the semi-discrete, finite degrees-of-freedom models on which hybrid simulations are based. Because these finite-degrees-of-freedom models are approximations of continuous real-world structures, information is lost and numerical errors are inevitably introduced into the hybrid simulation during the modeling process. Assumptions about the energy-dissipation and other analysis parameters further influence such errors.
- Other sources of numerical errors are the integration and equilibrium solution algorithms that are employed to solve the nonlinear equations of motion. This problem can be magnified if inappropriate algorithms and/or parameters for such algorithms (e.g., integration time-step size, algorithmic damping etc.) are selected for the hybrid simulation problem at hand. In addition, different integration schemes possess varying error propagation characteristics that can amplify experimental errors throughout a test.
- The third source of errors is the manner in which the transfer system imposes the calculated response quantities on the experimental subassemblies. As discussed earlier, continuous loading procedures generally achieve better accuracy than hold-and-ramp procedures because they eliminate force relaxation and actuator stick-slip errors. Furthermore, if the physical subassemblies exhibit velocity-dependent behavior, it is critical that the transfer system imposes the calculated response quantities in real time to minimize errors and obtain accurate results.
- Experimental errors generated by the transfer systems are the direst of all the errors affecting hybrid simulations. Poorly tuned transfer systems fail to accurately impose the calculated response quantities and introduce systematic errors such as undershoot, overshoot, lag, or feedback-loop instabilities. Furthermore, the interaction of the transfer system with its supporting reaction wall can lead to additional systematic errors. Because systematic errors modify the energy dissipation of a hybrid system, which in turn can render it unstable, they must be minimized or compensated for at all cost.
- Finally, the instrumentation devices and the data-acquisition system that measure the structural response can also introduce experimental errors. The resolution of the instrumentation, noise generated in the instrumentation and calibration errors in the data-acquisition system can produce very inaccurate or incorrect measurements that considerably affect a hybrid simulation and ultimately lead to poor test results.

2.8 SUMMARY

Hybrid simulation is one of the currently well-developed methods to conduct laboratory testing for evaluating the seismic behavior of structural systems and/or components thereof. In this experimental testing technique a simulation is executed based on a step-by-step numerical solution of the governing equations of motion for a model formulated considering both numerical and physical components of a structural system. Many of the advantages that are gained by utilizing hybrid simulations to perform experimental investigations were discussed and summarized in this chapter. In addition, a list of challenges that still need to be addressed in hybrid simulation was compiled as well. Afterwards, a brief history on the development of the testing technique from 1975 (when the first publication appeared) until 2008 (when this research was concluded) was presented. Subsequently it was shown that four key components are required in order to perform a hybrid simulation. These consist of a discrete model of the structure, including the static and dynamic loading; a transfer system, including a controller and static or dynamic actuators; the physical specimen that is being tested in the laboratory, including a support against which the actuators of the transfer system can react; and a data-acquisition system, including instrumentation devices. Because the finite element approach was utilized to introduce the basic concepts of a hybrid simulation, the testing procedure was next laid out in terms of the steps required to execute a direct integration analysis. In addition, several of the more common modes of employment of a hybrid simulation, such as local versus geographically distributed tests and slow versus rapid versus real-time tests, were discussed in detail. It was shown that a local hybrid simulation is defined by the collocation of all the required components in one laboratory. In contrast, for geographically distributed tests some or all the participating sites are dispersed and need to be connected through wide area networks. Depending on the execution speed of the test, the equations of motion that need to be solved during a hybrid simulation take on slightly different forms, which have been summarized and discussed. Finally, the chapter presented a short summary of numerical and experimental errors encountered in hybrid simulations and explained how such errors affect a test.

3 Object-Oriented Hybrid Simulation Framework

3.1 INTRODUCTION

Considerable research is currently being conducted worldwide to extend hybrid simulation to applications where advanced numerical techniques are utilized, boundary conditions are imposed in real time, and dynamic loading conditions are caused by seismic events, wind, blast, impact, waves, fire, and traffic. Similarly, efforts are under way to optimize capabilities for testing portions of a structure in geographically distributed laboratories, including techniques to improve network performance.

While only a few basic operations and communication protocols are needed to perform a hybrid simulation, to date few efforts (e.g., (Takahashi and Fenves 2006)) have been made to develop a common software framework for implementing and deploying systems that can carry out hybrid tests. Typically, each implementation and execution of hybrid simulation has been problem specific and used to be strongly dependent on the computational procedure, the configuration of the experiment, and the control and data-acquisition systems employed at the testing site. This means that hybrid simulations have generally been hard-coded for each specific problem at hand. Such highly customized software implementations are difficult to adapt to different structural problems and even harder to port to different laboratories. This is particularly true if multiple laboratories are involved in a hybrid simulation, as is the case for geographically distributed simulations. Thus, the lack of a common framework has posed a hurdle for those who considered utilizing hybrid simulation to investigate a structure experimentally; and additionally, has also limited collaboration among experts in the field.

A software framework is a reusable design for a system or subsystem and defines the overall architecture of such system, meaning its fundamental components as well as the relationships among them. Thus, a software framework for experimental testing should ideally support a large variety of computational software, structural testing methods, specimen types, testing configurations, control and data-acquisition systems, and communication protocols.

Furthermore, to accelerate the development and refinement of hybrid simulation, such software framework should be environment independent, robust, transparent, scalable, and easily extensible. It should allow domain researchers to execute hybrid simulations without specialized knowledge about the underlying software. At the same time, it should also enable hybrid simulation and IT specialists to extend the frontiers of the methodology by permitting the effortless addition of new developments.

Two different approaches are currently utilized when developing and implementing hybrid simulations. The first one is the structural analysis approach, in which a hybrid simulation can be viewed as a conventional finite element analysis where physical models of some portions of the structure are embedded in the numerical model. The second one is the control system approach where a hybrid simulation is treated as a feedback system with the computational portion of the structure acting as the controller and the physical portion resembling the plant. While it can be shown that the two approaches lead to equivalent algorithms (Sivaselvan 2006) both of them have advantages and disadvantages when investigating the properties and performance of hybrid simulations. For the development of the object-orient software framework discussed herein, the former finite element approach is the preferred method. Since in this approach the physical portions of a structure are embedded as subassemblies into the computational software, the experimental software framework should provide the means to interact with any finite element code that provides an application programming interface (API) and is not a black box (see Fig. 3.1). This enables users to choose their favorite finite element analysis software (with all the unmodified computational features) to perform a hybrid simulation, without the need to learn new analysis software.

Because the object-oriented software framework is designed to connect any finite element analysis software running on one or more machines with control and data-acquisition software running in one or more laboratories, it can be considered middleware. By definition middleware describes a piece of software that connects two or more software applications, allowing them to exchange data. It is similar to the middle layer of the three-tier architecture that is described in a later section. The Open-Source Framework for Experimental Setup and Control, OpenFresco, described herein, is a redesigned, improved, and extended version of the software framework originally presented by Takahashi and Fenves (2006), and provides in a convenient,

modular, and standardized manner those additional functions needed by analysis software to work with laboratory equipment to perform hybrid simulations.

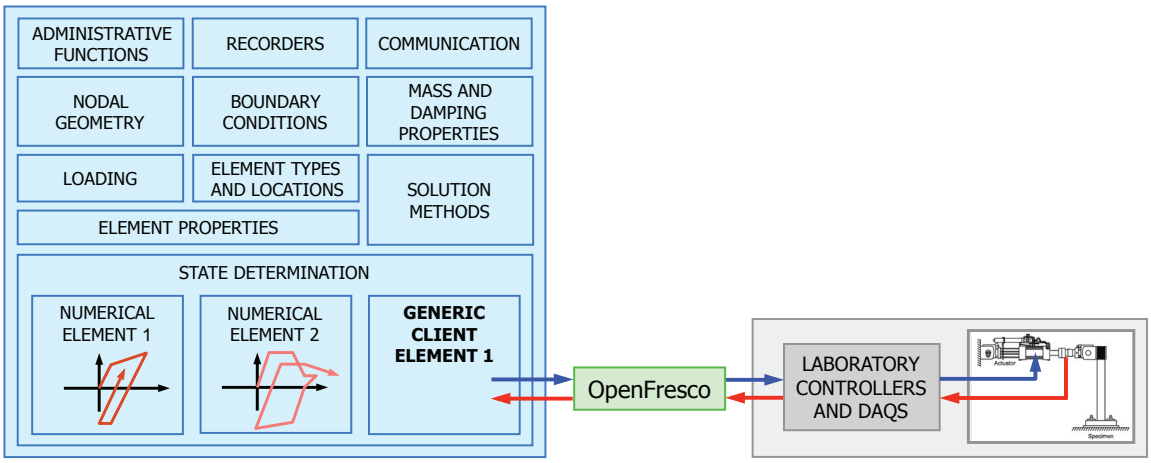


Fig. 3.1 Finite element analysis software of user's choice, OpenFresco middleware, and physical specimen with control and data-acquisition systems in laboratory.

This chapter first presents the fundamental building blocks (abstract classes) of the software framework (Takahashi and Fenves 2006) and explains how such objects were determined based on a rigorous systems analysis of the operations performed during hybrid simulations. The interaction among those classes and the underlying object-oriented software design patterns are explained as well. Thereafter, the concept of the multitier software architecture is discussed, which provides the means to interact with any finite element software and facilitates geographically distributed simulations. Communication methods and protocols necessary for distributed computation and testing are described next. Following this are brief descriptions on the implementation of the available objects (concrete sub-classes). Finally, the chapter explains how the Open System for Earthquake Engineering Simulation, OpenSees (McKenna 1997), can be used as the computational framework and the advantages attained by doing so.

3.2 OPENFRESCO SOFTWARE ARCHITECTURE

In compliance with object-oriented methodologies the first step in the development process of the software framework is to determine what a structural testing system is functionally required to do, to facilitate the execution of experiments. This first phase of the development process is

also called object-oriented analysis (OOA) and produces a conceptual model of the software framework. All three of the currently well-established and earlier described experimental methods (quasi-static testing, shaking table testing, and hybrid simulation) impose boundary conditions on a structure or components thereof and return the corresponding work conjugates. The boundary conditions that are to be imposed at the interface degrees of freedom can be prescribed displacements, tractions, or a combination. In structural testing such prescribed boundary conditions are generally applied by means of a transfer system, which consists of actuators and measuring devices. The manner in which the transfer systems are set up and physically connect to the specimen is test specific and therefore usually changes from experiment to experiment. Finally, the actuators need to be commanded by means of a control system and measurements need to be recorded by a data-acquisition system. These control and data-acquisition systems generally differ from laboratory to laboratory.

Hence, the main requirements for the software framework are (1) to provide some means to represent the structural subassemblies that are being tested experimentally; (2) to generate appropriate (displacements, velocities, accelerations, forces) input commands for the transfer systems; (3) to convert measured signals (displacements, velocities, accelerations, forces) back into appropriate forms for the computational software; and (4) to offer the flexibility to interact with a wide variety of control and data-acquisition systems. Additionally, the software framework should provide (5) techniques to distribute processes across a network to different machines, thereby enabling geographically distributed simulations; (6) means to transmit additional messages between the computational software and the laboratory; and (7) execution and communication speeds that make real-time testing possible.

The next step in the object-oriented analysis process is to generate an object-oriented model (OOM) by breaking down the previously identified requirements into components. This means that abstractions of the real-world objects are generated in the software. These abstract components, which are blueprints describing the characteristics and behavior of a real-world object, are called software classes. Using encapsulation, a class conceals its functional details (data and operations on data) from other objects that send messages to it and thereby provides the basis for modular, flexible, and extensible software. According to the previously explained components and requirements necessary to provide a bridge between a standard finite element

analysis program and laboratory control and data-acquisition systems, the following software abstractions are introduced.

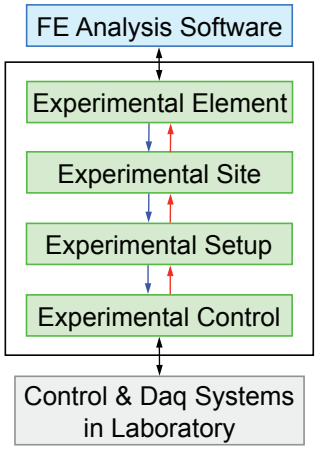


Fig. 3.2 Abstract components of the OpenFresco software architecture.

Experimental Element:

The abstraction of the first of the previously described real-world components is the ExperimentalElement class. It represents specimens such as trusses, beams, columns, braces, springs, walls, and other parts of a structure that are physically tested in a laboratory. Analogous to an analytical element, the experimental element acts within the host finite element analysis software, but represents a physical portion of a model instead of a numerical one. Like the analytical element the basic functionality of an experimental element is to provide the mass, damping, and stiffness matrices as well as the residual force vector for a given deformation state. The experimental element is thus responsible for transforming prescribed boundary conditions from the global coordinate system of the FE-model into the local or basic element coordinate system that is best suited for testing. Additionally, it needs to provide inverse transformations for the measured response quantities, such as displacements, velocities, accelerations and forces, from the element coordinate system back to the global coordinate system of the FE-model. Contrary to analytical elements, the experimental element is oftentimes unable to return a tangent stiffness matrix and needs to return its initial stiffness matrix instead. This is because it is generally difficult to compute an accurate element tangent stiffness matrix solely from the experimentally measured response quantities at the basic or local element degrees of freedom. Furthermore, since a physical specimen is truly history dependent, the experimental element representing such specimen cannot revert to a previous or initial state of an analysis once a

simulation is in progress. The *ExperimentalElement* class is an abstract class, which defines the interface that all experimental elements must conform to. The concrete subclasses of the *ExperimentalElement* class then provide the implementations for the specific experimental specimens.

Experimental Site:

Other real-world components involved in an experimental test are the laboratories in which the testing takes place and the computational sites where the analysis is carried out. The *ExperimentalSite* class represents such laboratories and computational sites in the software framework. Because in the real world, laboratories and computational sites can be in different places, the experimental site objects in the software are responsible for providing communication methods to connect the different sites and store data that are received from or need to be sent to other sites. Furthermore, the experimental sites should support different communication protocols such as TCP/IP, UDP, or NHCP (NEES hybrid-simulation control protocol) and be able to guarantee secure transactions over the Internet using cryptographic protocols such as TLS (transport layer security) or SSL (secure sockets layer). The *ExperimentalSite* class is an abstract base class, which defines the interface that all experimental sites must conform to. The concrete subclasses of the *ExperimentalSite* class then provide the implementations for the specific laboratories and computational sites.

Experimental Setup:

The third abstraction is the *ExperimentalSetup* class, the software class that is responsible for representing the possible configurations of the transfer systems constructed in the laboratories. Since a transfer system consists of actuators and measuring devices, the experimental setup needs to transform the prescribed boundary conditions from the local or basic element degrees of freedom of the experimental elements into the actuator degrees of freedom of the transfer system, utilizing the geometry and the kinematics of the loading and instrumentation system. Similarly the work conjugates measured by transducers and load cells need to be transformed back to the experimental element degrees of freedom. These transformations can be implemented either as simple linear transformation matrices (small displacements) or as complex algebraic transformations taking large-displacement effects into account. The *ExperimentalSetup* class is again an abstract base class, which defines the interface that all experimental setups must

conform to. The concrete subclasses of the *ExperimentalSetup* class then provide the implementations for the specific setups.

Experimental Control:

The last one of the main abstractions introduced here is the *ExperimentalControl* class. Since one of the required tasks for experimental testing is the communication with a wide variety of control and data-acquisition systems, the *ExperimentalControl* class provides such interface and functionality in the software framework. The advantage of this abstraction and the encapsulation of these operations is that the *ExperimentalSetup* class separates the details of the loading system configuration from the *ExperimentalControl* class. Thus, those responsible for the IT aspects of the control and data-acquisition systems need be concerned only with the *ExperimentalControl* class; while those configuring the actuators and sensors focus on the *ExperimentalSetup* class. As with the other classes, the *ExperimentalControl* class is again an abstract base class, which defines the interface that all experimental controls must conform to. The concrete subclasses of the *ExperimentalControl* class then provide the implementations for the specific interfaces with different control and data-acquisition systems.

3.3 MULTI-TIER SOFTWARE ARCHITECTURE

While the components described in the previous section build the core of the software framework, they do not provide a modular and flexible way to utilize OpenFresco with any finite element analysis software of the user's choice. To achieve this objective, the three- and multi-tier software architectures are introduced. They both belong to the class of client/server architectures, which is based on passing messages among different software agents. The three-tier software architecture wraps around the OpenFresco core modules and provides the necessary functionality to interface with a wide variety of finite element software clients and control and data-acquisition systems.

As the name suggests the software is divided into three layers composed of a client, a middle-tier (or application) server, and a backend server. The middle-tier server is located between the user interface (client) and the data management (server) components. In the traditional three-tier architecture this middle-tier provides functional process management where business logic and rules are executed. The three-tier architecture is typically employed whenever

a distributed client/server design is required that provides increased performance, flexibility, maintainability, reusability, and scalability while hiding the complexity of distributed processing from the user (Carnegie Mellon SEI).

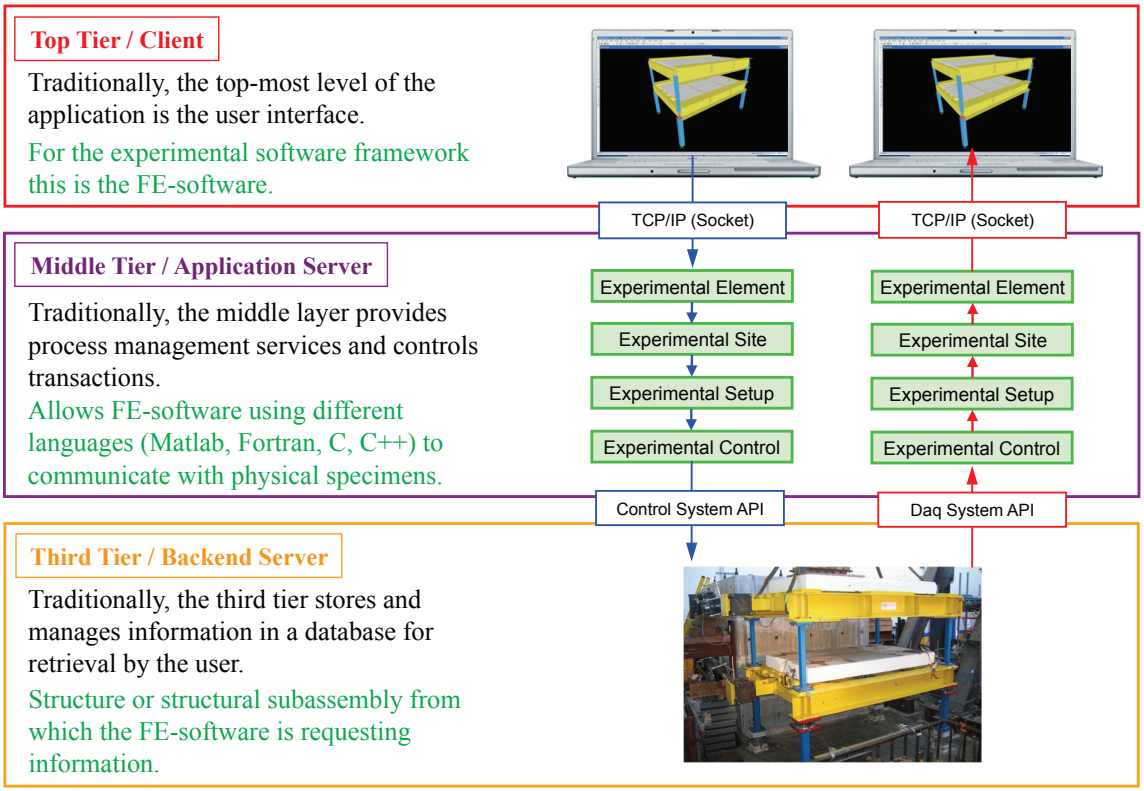


Fig. 3.3 Multi-tier software architecture.

As can be seen from Fig. 3.3, in the case of the experimental software framework, the finite element analysis software resembles the client or user-interface top tier, where the structural model is created and then analyzed. The backend server, also called third tier, is the laboratory server composed of the different control and data-acquisition systems with their APIs. This means that comparable to the general three-tier architecture, where the third tier is typically a database from which a user is querying information, the laboratory server can be seen as the structure or structural subassembly from which the finite element analysis software is querying information. Finally, the middle-tier server, which is also called simulation application server, provides process management services and data transformations so that different computational clients can query information from the different experimentally tested portions of a structure. It improves performance, flexibility, maintainability, reusability, and scalability by centralizing process logic. Centralized process logic makes administration and change management easier by

localizing system functionality so that changes must be written only once and placed on the middle-tier server to be available throughout the systems (Carnegie Mellon SEI). In addition, due to this separation provided by the middle-tier server, finite element analysis software written in different languages such as Matlab, Fortran, C, C++, etc., can be accommodated. If the middle-tier server is further divided into two or more units, the design is referred to as multi- or n-tier architecture, which is one of the architectural software patterns. As will be seen shortly, for geographically distributed simulations OpenFresco is employing a four-tier architecture with two middle-tier server units.

Because of the modularity and flexibility of the multi-tier architecture combined with the OpenFresco middle-tier components, a wide variety of local and geographically distributed testing configurations are possible. For all these configurations the client and the first middle-tier server unit will usually be running on the same machine. However, since TCP/IP sockets are used to achieve the interprocess communications between the two, the client and the first simulation application server unit could also be run on two separate machines, if it is desired. On the other hand, the communications between the two simulation application server software agents are always distributed, which means that the processes are running on two different machines. These network communications are managed by the previously described *ExperimentalSite* class, which in turn is relying on the OpenSees *Channel* class. The OpenSees *Channel* class can then use different communication protocols such as TCP/IP, UDP, or NHCP.

In the case of a local deployment of OpenFresco, meaning that all processes are executed locally in one laboratory, there are two possible configurations available as shown in Fig. 3.4. In the first configuration, which is the more general one, a generic client element has to be added to the finite element software that is being used for the analysis. The generic client element requires only the number of nodes, the degrees of freedom it is connected to, and the port to communicate through as input parameters. Furthermore, it can be implemented in the language of the corresponding finite element analysis software. Thus, data exchange with the OpenFresco middleware takes place through the generic client element, embedded by the user into each finite element analysis program using their published programming interfaces (e.g., user-defined elements). Utilizing the socket provided by the OpenFresco software framework, such user-defined element communicates with the simulation application element server through a TCP/IP connection. Since a generic element is used in the finite element analysis software, the

simulation application element server then interfaces with the OpenFresco experimental element class. The advantage of this first configuration is that only one element, the generic client element, has to be added to the finite element analysis software and all the existing experimental elements in the OpenFresco software framework can be utilized to represent the experimental portions of a structure.

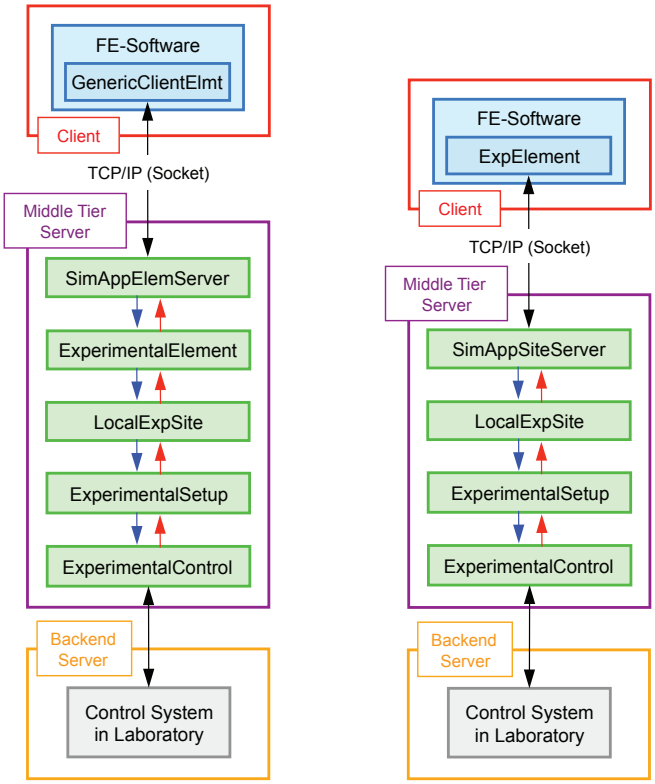


Fig. 3.4 Three-tier configurations for local deployment.

In the second configuration in Fig. 3.4, an experimental element instead of a generic client element is directly added to the finite element analysis software. This experimental element then communicates with the simulation application site server in the OpenFresco software framework, which in turn interfaces with the local experimental site instead of the experimental element. The advantage of the second configuration is, that if a very specialized experimental element is needed, which is not available in and cannot be added to OpenFresco (because it utilizes some resources available only within the finite element analysis software), such experimental element can directly be added to the computational software client.

As can be seen from Fig. 3.5, in the case of a distributed deployment of OpenFresco, two very similar possible configurations are available. Their main difference is that the middle-tier

server is divided into two units that are distributed across a network. The shadow and actor experimental sites, which are concrete subclasses of the *ExperimentalSite* base class, manage the communications between the two tiers. As mentioned before, they make use of the OpenSees *Channel* class and can therefore employ different protocols such as TCP/IP, TCP/IP with SSL, UDP, or NHCP for communications.

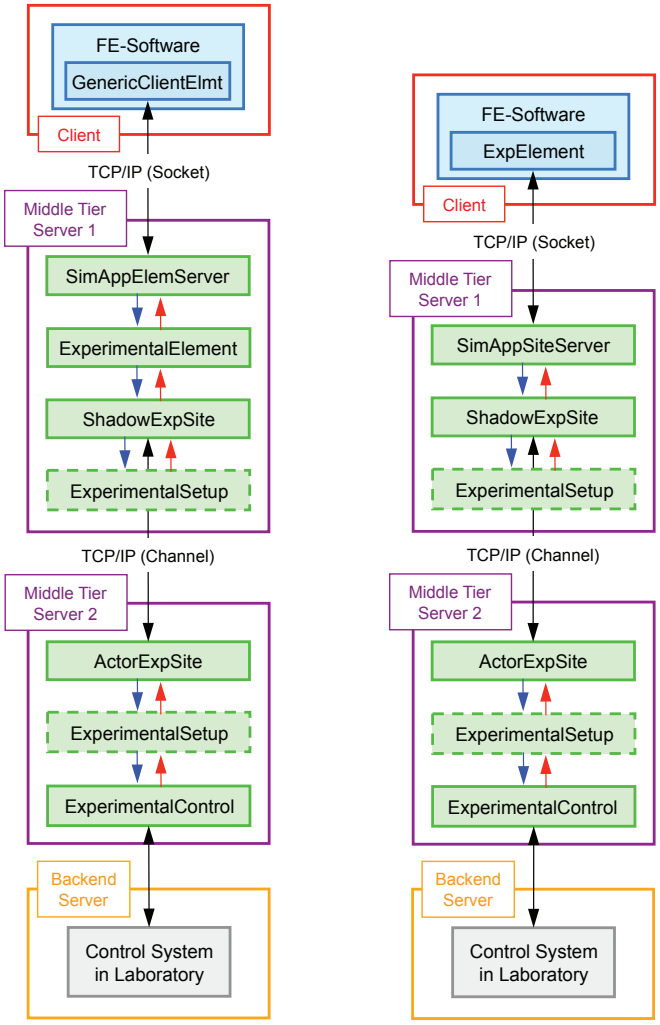


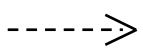
Fig. 3.5 Four-tier configurations for network deployment.

Because the middle-tier server is divided into two units, the *ExperimentalSetup* class, which is responsible for transforming between element and actuator degrees of freedom, can be located either on the first middle-tier server unit (usually running on the same machine as the finite element analysis client) or on the second middle-tier server unit (usually running in the laboratory), but not on both sides simultaneously. The purpose of providing this flexibility to the user is to accommodate a finite element analyst's view of a hybrid simulation as well as an

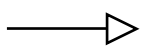
experimentalist's. The former is typically interested in having some finite elements that return resisting forces provided some trial response, without having to deal with experimental setups in a laboratory. In such cases the experimental setup can be placed on the laboratory side. The latter, on the other hand, generally prefers to directly receive commands and return measurements in the control and data-acquisition system degrees of freedom without having to transform from and to element degrees of freedom. In this case the experimental setup can conveniently be placed on the side of the computational client. So to summarize, OpenFresco's three- and four-tier architectural patterns provide the modularity and flexibility to interface any computational agent running on one or more machines with physical specimens being tested in one or more laboratories.

3.4 OBJECT-ORIENTED DESIGN DETAILS

In the next step of the object-oriented methodology, the object-oriented design (OOD) phase, the interfaces of the objects and the interactions amongst them are defined. The basis for the design phase is the conceptual model and all the possible configurations (use cases) of the software framework that were developed during the object-oriented analysis in the previous subsections. Fig. 3.6 shows the class diagram of the OpenFresco software framework using simplified graphical notation of the Unified Modeling Language (UML) (Booch et al. 2005). The classes are graphically represented by a rectangle with the name of the class inside the rectangle. For abstract classes the name is displayed in Italic. The attributes and operations of a class are omitted in the diagram presented here and are explained separately afterwards. The following four types of lines represent the relationships among the classes:

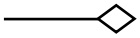


A *dependency* is a relationship that states that one class uses operations, variables or arguments from another class. The relationship is sometimes also referred to as "using-a" relationship.



A *generalization* is a relationship between a parent (superclass or base class) and a child (subclass or derived class). The relationship is also called inheritance or is referred to as "is-a" relationship. A child inherits the properties, such as attributes and operations, from its parent class.

An *association* is a structural relationship that specifies that objects of one class are connected to objects of an other class. The relationship is also referred to as "knows-a" relationship.



An *aggregation* is a special association relationship that specifies that an object of one class is made up of objects of other classes. The relationship is also referred to as "has-a" relationship.

The multiplicity of an association or aggregation specifies how many objects may be connected across an instance. It is written as a range of integers with the minimal and maximal values and displayed at the ends of an association or aggregation.

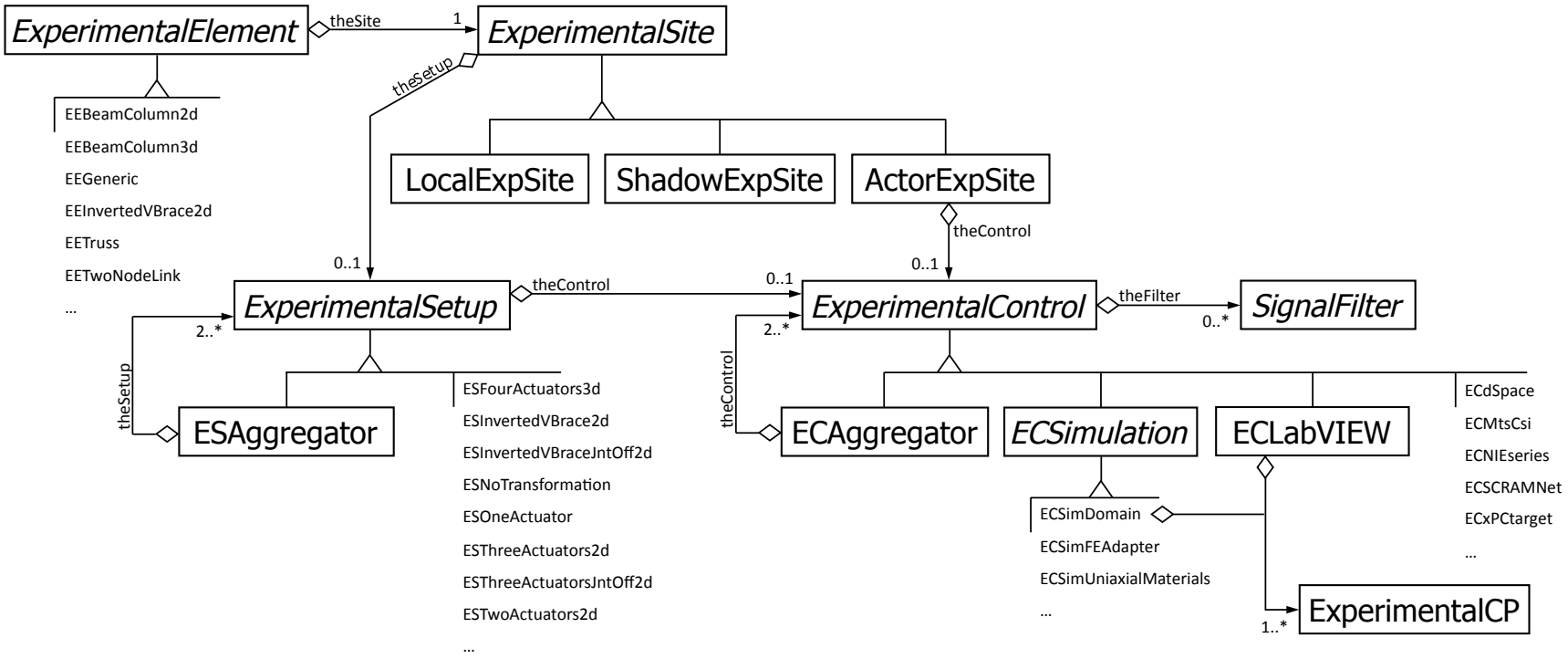


Fig. 3.6 Class diagram of the OpenFresco software framework.

Experimental Element:

As can be seen from the class diagram in Fig. 3.6, the *ExperimentalElement* class is an abstract class that inherits attributes and operations from the *Element* class, which is an abstract base class in OpenSees.

```
class ExperimentalElement : public Element {
public:
    // constructor and destructor
    ExperimentalElement(int tag, int classTag,
        ExperimentalSite *site = 0);
    virtual ~ExperimentalElement();

    // public method to obtain basic DOF size which is
    // equal to the max num DOF that can be controlled
    virtual int getNumBasicDOF() = 0;

    // public methods to set and to obtain the stiffness
    virtual int setInitialStiff(const Matrix& stiff) = 0;
    const Matrix &getTangentStiff();
    const Matrix &getInitialStiff();

    // public methods to obtain the daq response in global system
    virtual const Vector &getDisp();
    virtual const Vector &getVel();
    virtual const Vector &getAccel();
    virtual const Vector &getTime();

protected:
    // pointer to ExperimentalSite
    ExperimentalSite* theSite;

    // size of ctrl/daq data
    // [0]:disp, [1]:vel, [2]:accel, [3]:force, [4]:time
    ID* sizeCtrl;
    ID* sizeDaq;

    // initial stiffness matrix
    Matrix theInitStiff;

private:
    // the following methods must be defined if the Element object
    // is inherited, but are meaningless for this element.
    int revertToLastCommit();
    int revertToStart();

    bool firstWarning;
};
```

Fig. 3.7 Interface for the ExperimentalElement class.

The implementations of the experimental elements, such as truss, beam, column, brace, spring, and others are provided in the concrete subclasses of the *ExperimentalElement* class. The class interface is shown in Fig. 3.7 and provides the following modified and additional functionalities as compared to its parent *Element* class. First of all, and most importantly, the

ExperimentalElement class is associated with the abstract *ExperimentalSite* class. More specifically, the relationship between these two classes is an aggregation, meaning that an experimental element has one experimental site. Furthermore, a method for setting the initial stiffness matrix of the experimental element and several methods for obtaining the measured response quantities have been added to the interface. Most of the methods defined in the *ExperimentalElement* interface are pure virtual, meaning that the concrete subclasses need to provide the implementations. Finally, since a physical specimen is truly history dependent and an experimental element representing such specimen cannot revert to a previous or initial state of an analysis, the two inherited methods `revertToLastCommit()` and `revertToStart()` return an error message and interrupt the simulation.

Experimental Site:

The redesigned *ExperimentalSite* class is an abstract base class with the interface defined, as shown in Fig. 3.8. It has three derived classes, which provide the concrete implementations necessary for local and geographically distributed simulations. The LocalExpSite class is employed for local testing and the ShadowExpSite and ActorExpSite classes are employed as a pair for distributed testing. In terms of the terminology generally used for client-server architectures, the ShadowExpSite provides the implementation of the client and the ActorExpSite the one of the server.

```
class ExperimentalSite : public TaggedObject {
public:
    // constructors and destructor
    ExperimentalSite(int tag,
        ExperimentalSetup *setup = 0);
    ExperimentalSite(const ExperimentalSite &site);
    virtual ~ExperimentalSite();

    virtual int setSize(ID sizeT, ID sizeO);
    virtual int setup() = 0;

    // public methods to set and to get responses
    virtual int setTrialResponse(const Vector* disp,
        const Vector* vel,
        const Vector* accel,
        const Vector* force,
        const Vector* time);
    virtual int setTrialDisp(const Vector* disp);
    virtual int getDaqResponse(Vector* disp,
        Vector* vel,
        Vector* accel,
        Vector* force,
        Vector* time);
                                                    (continued)
```

```
    virtual int setDaqResponse(const Vector* disp,
        const Vector* vel,
        const Vector* accel,
        const Vector* force,
        const Vector* time);
    virtual int checkDaqResponse() = 0;

    virtual const Vector& getTrialDisp();
    virtual const Vector& getTrialVel();
    virtual const Vector& getTrialAccel();
    virtual const Vector& getTrialForce();
    virtual const Vector& getTrialTime();

    virtual const Vector& getDisp();
    virtual const Vector& getVel();
    virtual const Vector& getAccel();
    virtual const Vector& getForce();
    virtual const Vector& getTime();

    virtual int commitState();

    virtual ExperimentalSite *getCopy() = 0;

    virtual int getTrialSize(int rType);
    virtual int getOutSize(int rType);
    virtual int getCtrlSize(int rType);
    virtual int getDaqSize(int rType);

protected:
    ...
```

Fig. 3.8 Interface for the ExperimentalSite class.

The ShadowExpSite and the ActorExpSite should support different communication protocols such as TCP/IP, UDP, or NHCP (NEES hybrid-simulation control protocol) and be able to guarantee secure transactions over the Internet using cryptographic protocols such as TLS (Transport Layer Security) or SSL (Secure Sockets Layer). Since the distributed-computing model developed in OpenSees provides most of these requirements, the ShadowExpSite and ActorExpSite classes are derived from the *Shadow* and *Actor* classes of such model, respectively. The *Shadow* and *Actor* classes are both associated with the OpenSees *Channel* class, which is an abstract base class that has concrete subclasses implementing the different communication protocols. The available child classes useful for distributed testing are TCP_Socket, UDP_Socket and TCP_SocketSSL.

Experimental Setup:

The interface of the redesigned *ExperimentalSetup* abstract base class is defined as shown in Fig. 3.9. Similar to the other abstract classes most of the methods in the *ExperimentalSetup* class are pure virtual, and the concrete subclasses provide the implementations for the different

experimental setups. As explained before, the Bridge software pattern is used between the *ExperimentalSetup* class and the associated *ExperimentalControl* class. Furthermore, the class interface provides methods for setting and getting trial and data-acquisition response quantities at and from the *ExperimentalSite* and *ExperimentalControl* classes.

The transformation of the prescribed boundary conditions from the local or basic element degrees of freedom of the experimental elements into the actuator degrees of freedom of the transfer system, which is the first core task of the *ExperimentalSetup* class, is performed by the public `transfTrialResponse()` method. Similarly, the transformation of the work conjugates measured by transducers and load cells back to the experimental element degrees of freedom, which is the second core task of the *ExperimentalSetup* class, is performed by the public `transfDaqResponse()` method. Both of these methods then call a sequence of private methods from the concrete subclasses to transform each individual response quantity in turn. The remaining public methods are to set factors that are utilized to modify the control and data-acquisition values in order to account for unit conversions and similitude transformations.

```
class ExperimentalSetup : public TaggedObject {
public:
    // constructors and destructor
    ExperimentalSetup(int tag,
        ExperimentalControl* control = 0);
    ExperimentalSetup(const ExperimentalSetup& es);
    virtual ~ExperimentalSetup();

    virtual int setSize(ID sizeT, ID sizeO) = 0;
    virtual int setup() = 0;
    virtual int setCtrlDaqSize();

    // public methods to set and to obtain responses
    virtual int setTrialResponse(const Vector* disp,
        const Vector* vel,
        const Vector* accel,
        const Vector* force,
        const Vector* time);
    virtual int getTrialResponse(Vector* disp,
        Vector* vel,
        Vector* accel,
        Vector* force,
        Vector* time);
    virtual int setDaqResponse(const Vector* disp,
        const Vector* vel,
        const Vector* accel,
        const Vector* force,
        const Vector* time);
    virtual int getDaqResponse(Vector* disp,
        Vector* vel,
        Vector* accel,
        Vector* force,
        Vector* time);
                                                    (continued)
```

```
    // public methods to transform the responses
    virtual int transfTrialResponse(const Vector* disp,
        const Vector* vel,
        const Vector* accel,
        const Vector* force,
        const Vector* time);
    virtual int transfDaqResponse(Vector* disp,
        Vector* vel,
        Vector* accel,
        Vector* force,
        Vector* time);

    virtual int commitState();

    virtual ExperimentalSetup *getCopy() = 0;

    // public methods to set the control and daq factors
    void setCtrlDispFactor(const Vector& f);
    void setCtrlVelFactor(const Vector& f);
    void setCtrlAccelFactor(const Vector& f);
    void setCtrlForceFactor(const Vector& f);
    void setCtrlTimeFactor(const Vector& f);

    void setDaqDispFactor(const Vector& f);
    void setDaqVelFactor(const Vector& f);
    void setDaqAccelFactor(const Vector& f);
    void setDaqForceFactor(const Vector& f);
    void setDaqTimeFactor(const Vector& f);

    virtual ID getCtrlSize();
    virtual ID getDaqSize();
    virtual int getCtrlSize(int rType);
    virtual int getDaqSize(int rType);

protected:
    ...
```

Fig. 3.9 Interface for the ExperimentalSetup class.

As can be seen from Fig. 3.6, the experimental setup hierarchy provides an ESAggregator subclass that is associated with its own *ExperimentalSetup* parent class. This allows two or more concrete experimental setups to be aggregated into a single experimental setup, which then interacts with the *ExperimentalSite* and *ExperimentalControl* objects. This structural pattern is known as Composite pattern and provides the means for the *ExperimentalSite* class to treat individual experimental setups and compositions of experimental setups uniformly (Gamma et al. 1995).

Experimental Control:

The redesigned *ExperimentalControl* class is an abstract base class with the interface defined as shown in Fig. 3.10. Once more, most of the methods in the abstract class interface are defined as pure virtual, and the concrete subclasses are responsible to supply the implementations. The

public `setTrialResponse()` and `getDaqResponse()` methods, which depend on all the response quantities, call the protected `control()` and `acquire()` methods. The protected methods then utilize the application programming interfaces (APIs) of the control and data-acquisition systems in order to interact with such systems. This approach provides the flexibility to only use and return the response quantities supported by the specific APIs.

Similar to the *ExperimentalSetup* hierarchy the Composite software pattern is employed again to provide an ECAggregator subclass that allows two or more experimental control objects to be aggregated into one composite. This enables the software framework to utilize one interface for controlling the actuators while a second one is utilized for the data acquisition.

To analytically model control and data-acquisition systems and thereby provide software capabilities to simulate an experimental test before its actual execution, the *ECSimulation* class is added to the hierarchy. This is particularly useful for checking a hybrid model and estimating the response of a structure before running an actual test. In addition, the ECSimFEAdapter subclass provides the means to simulate parts of a structure by different finite element analysis software. This ability to couple finite element analysis software provides additional flexibility and greater realism in simulating large engineering systems than may be possible with a single program. As can be seen from Fig. 3.6 the *ECSimulation* class is an abstract class that is derived from the *ExperimentalControl* class. Furthermore, the *ExperimentalControl* class is associated with the abstract *SignalFilter* class that can be employed to filter measured signals or simulate experimental errors. Finally, the ExperimentalCP class is used to define control points, which are required by some of the concrete experimental control subclasses. A control point is a logical container of one or more directions (degrees of freedom) for sending command signals to a control system or acquiring feedback signals from a data-acquisition system.

```
class ExperimentalControl : public TaggedObject {
public:
    // constructors and destructor
    ExperimentalControl(int tag);
    ExperimentalControl(const ExperimentalControl& ec);
    virtual ~ExperimentalControl();

    virtual int setSize(ID sizeT, ID sizeO) = 0;
    virtual int setup() = 0;

    // public methods to set and to get response
    virtual int setTrialResponse(const Vector* disp,
        const Vector* vel,
        const Vector* accel,
        const Vector* force,
        const Vector* time) = 0;
                                                    (continued)
```

```
    virtual int getDaqResponse(Vector* disp,
        Vector* vel,
        Vector* accel,
        Vector* force,
        Vector* time) = 0;

    virtual int commitState();
    virtual ExperimentalControl *getCopy() = 0;

    const ID& getSizeCtrl();
    const ID& getSizeDaq();

protected:
    // protected methods to set and to get response
    virtual int control() = 0;
    virtual int acquire() = 0;

    // size of ctrl/daq data
    // [0]:disp, [1]:vel, [2]:accel, [3]:force, [4]:time
    ID *sizeCtrl;
    ID *sizeDaq;

    // signal filter
    SignalFilter *theFilter;
};
```

Fig. 3.10 Interface for the ExperimentalControl class.

Interactions and Transformations:

The flow of data between the OpenFresco modules and the transformation of the data within such modules is illustrated by the two-dimensional beam-column example shown in Fig. 3.11. The following terminology is used for the data flow: Data that are flowing from the finite element analysis software towards the laboratory are called (1) “trial data” (if entering an object/class) or (2) “ctrl data” (if leaving an object/class); and data that are flowing from the laboratory towards the finite element analysis software are called (3) “daq data” (if entering an object/class) or (4) “out data” (if leaving an object/class).

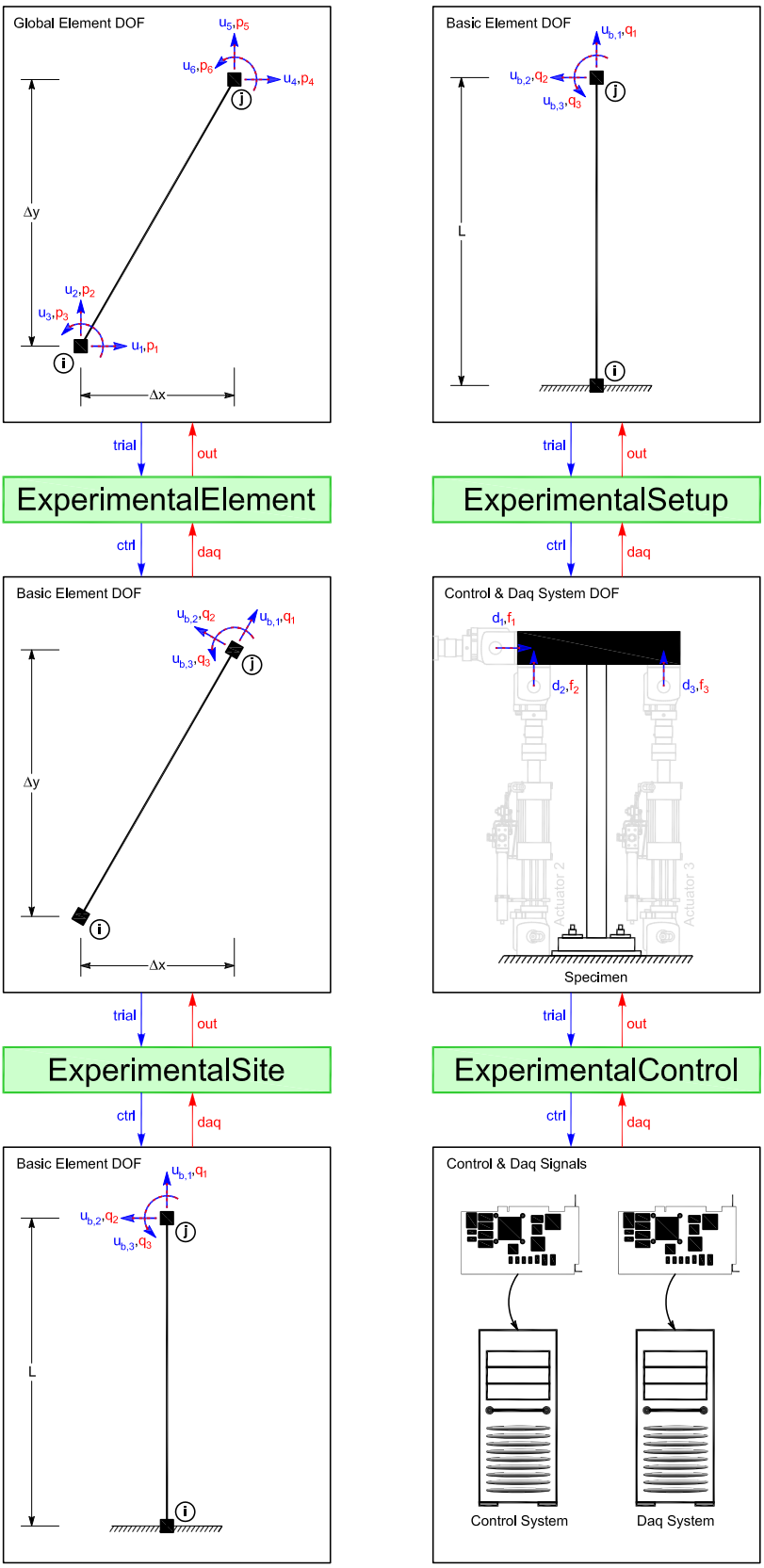


Fig. 3.11 Data flow and transformations for beam-column example.

As can be seen from the first picture in Fig. 3.11, the 2D beam-column element has a total of six global degrees of freedom, where $\mathbf{u}$ are the global element displacements and $\mathbf{p}$ are the global element forces. The ExperimentalElement then transforms these global trial displacements to control displacements (respectively deformations) in the cantilever basic coordinate system, $\mathbf{u_b}$. The next module, the ExperimentalSite, stores the trial data and, in case of a distributed test, reorganizes such data to facilitate a more efficient transfer across the network. Next, the three basic trial displacements/deformations, $\mathbf{u_b}$, are transformed to actuator control displacements, $\mathbf{d}$, by the ExperimentalSetup object. Finally, the ExperimentalControl module converts the actuator trial displacements into control signals that are appropriate for the employed laboratory control system. On the way back, the force signals measured by the data-acquisition system are converted into an output vector of forces by the ExperimentalControl object. Afterwards, the ExperimentalSetup module transforms these forces, $\mathbf{f}$, from the load cell (respectively data-acquisition system) degrees of freedom into basic element forces, $\mathbf{q}$. In case of a distributed simulation, the data are again reorganized into a structure that facilitates more efficient network transactions and are subsequently stored by the ExperimentalSite object for repeated retrieval, avoiding further network communications. Finally, the ExperimentalElement transforms the daq forces, $\mathbf{q}$, from the cantilever basic coordinate system into global element forces, $\mathbf{p}$, which are used by the computational code to assemble the global resisting force vector.

Sequence Diagrams for Local and Distributed Simulations:

The interaction diagrams that are shown next use the Unified Modeling Language (UML) to illustrate the sequence of messages that are sent to and received from objects.

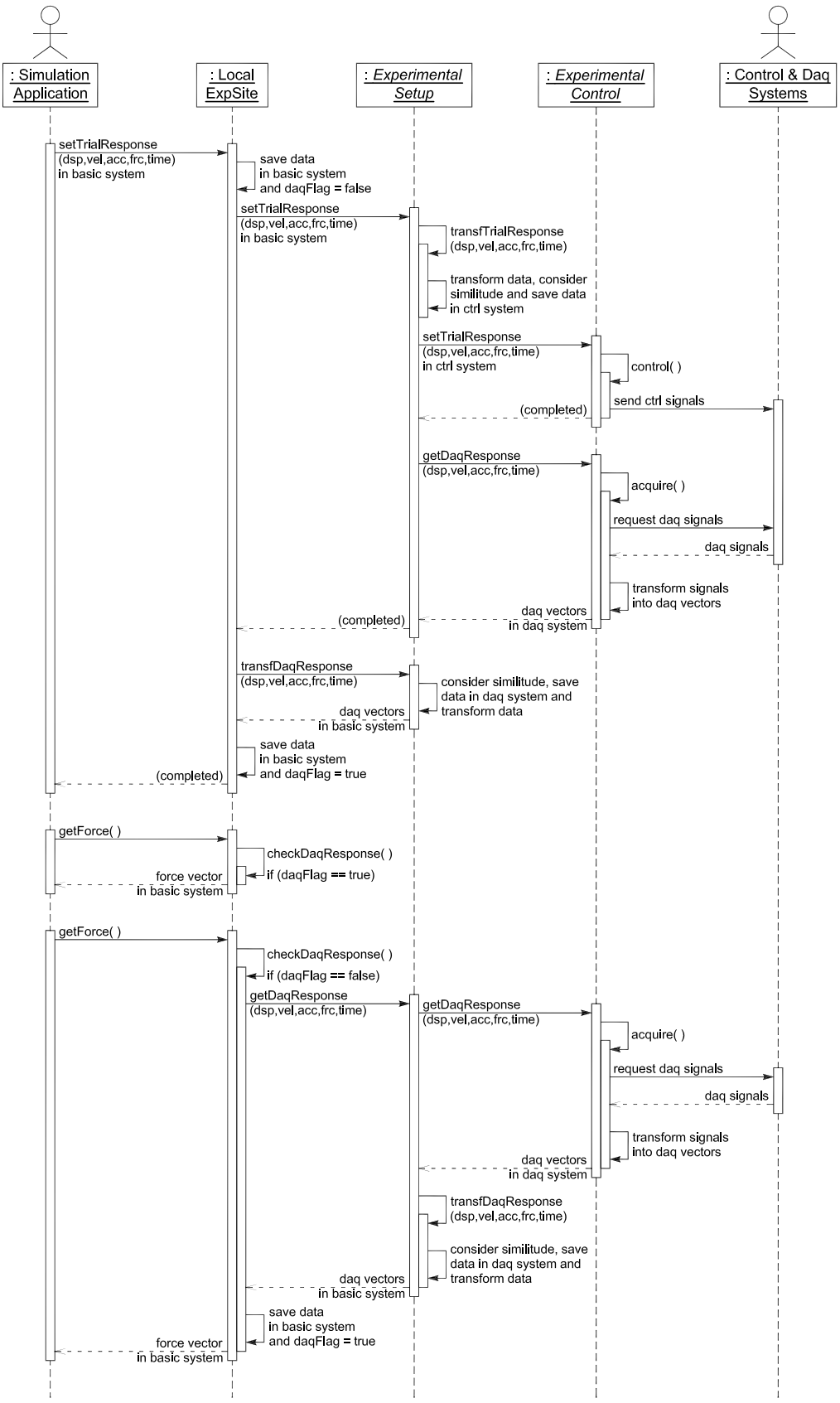


Fig. 3.12 OpenFresco sequence diagram for local deployment.

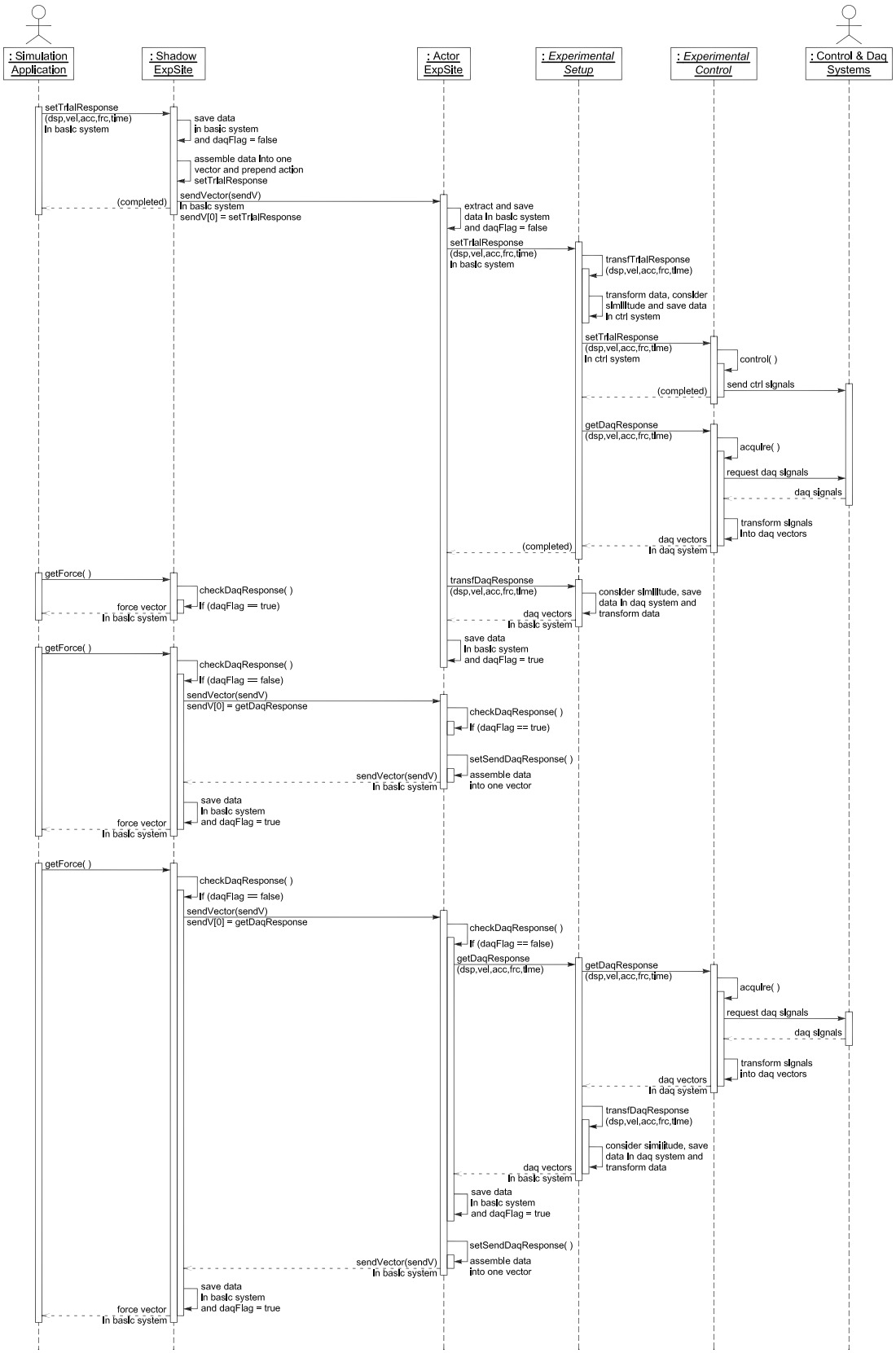


Fig. 3.13 OpenFresco sequence diagram for distributed deployment with xperimentalSetup on laboratory side.

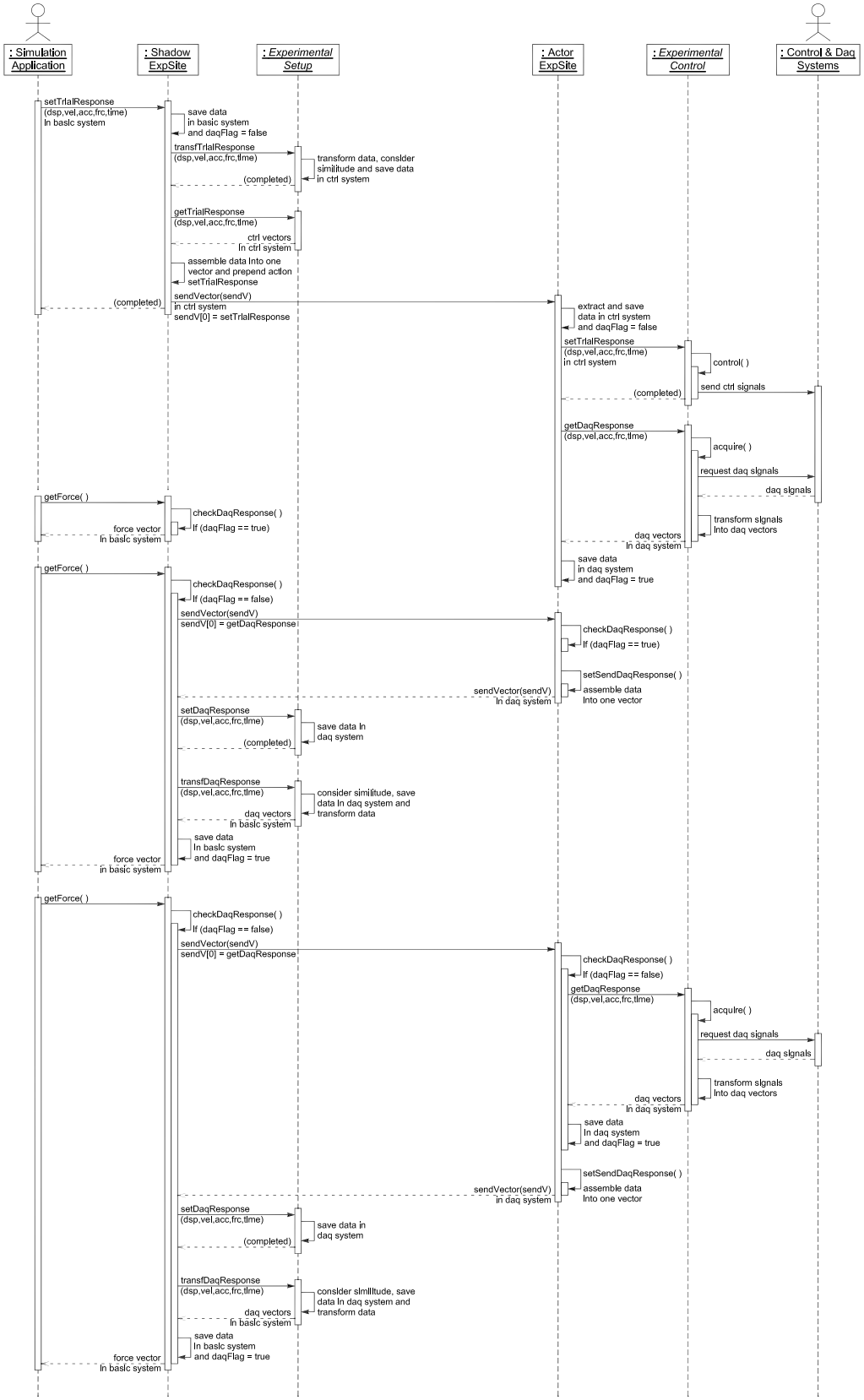


Fig. 3.14 OpenFresco sequence diagram for distributed deployment with ExperimentalSetup on computational client side.

3.5 CONCRETE SUBCLASSES

The currently implemented and available concrete sub-classes are explained next. These libraries of concrete OpenFresco classes realize the following two important objectives: (1) they offer domain researchers the opportunity to begin using the software framework without much effort and (2) they provide a convenient starting point for developers to share implementations. The existing concrete subclasses can serve as templates when implementing new concrete objects for the OpenFresco software framework. To simplify the use of the software framework an interpreter has been implemented using the Tcl Scripting Language (Tcl Developer Xchange). The OpenFresco interpreter extends the Tcl interpreter by adding commands for modeling (using nodes and elements) and for generating instances of site, setup, and control objects. It should be understood that with appropriate realizations of *ExperimentalElement*, *ExperimentalSite*, *ExperimentalSetup* and *ExperimentalControl* objects, portions of the overall hybrid structure can be modeled and simulated numerically using the special capabilities of different finite element analysis software packages.

3.5.1 Experimental Elements

Truss:

The experimental truss element has one axial degree of freedom, is defined by two nodes and can be used in 1D-, 2D-, and 3D-problems.

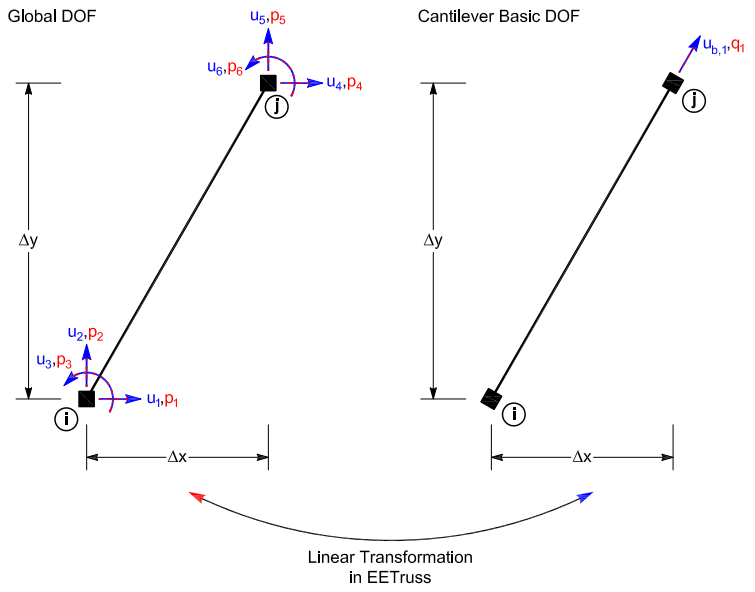


Fig. 3.15 Experimental truss element (EETruss).

The transformations of trial displacements and resisting forces from the 6 or 12 degrees of freedom in the global coordinate system to the one axial degree of freedom in the cantilever basic coordinate system and back are treated as linear geometric transformations and are implemented in the EETruss class. As can be seen from Fig. 3.15, for the experimental truss element, the degree of freedom at which the displacement is controlled (blue) and the one at which the resisting force is acquired (red), are collocated.

Beam-Column:

Depending on the dimension of the problem (2D or 3D), the experimental beam-column element formulation is based on the 3 or 6 collocated degrees of freedom of the cantilever basic system. The necessary transformations of displacements, velocities, accelerations and forces are implemented in the EEBeamColumn2d class and EEBeamColumn3d class for the 2D and 3D cases, respectively.

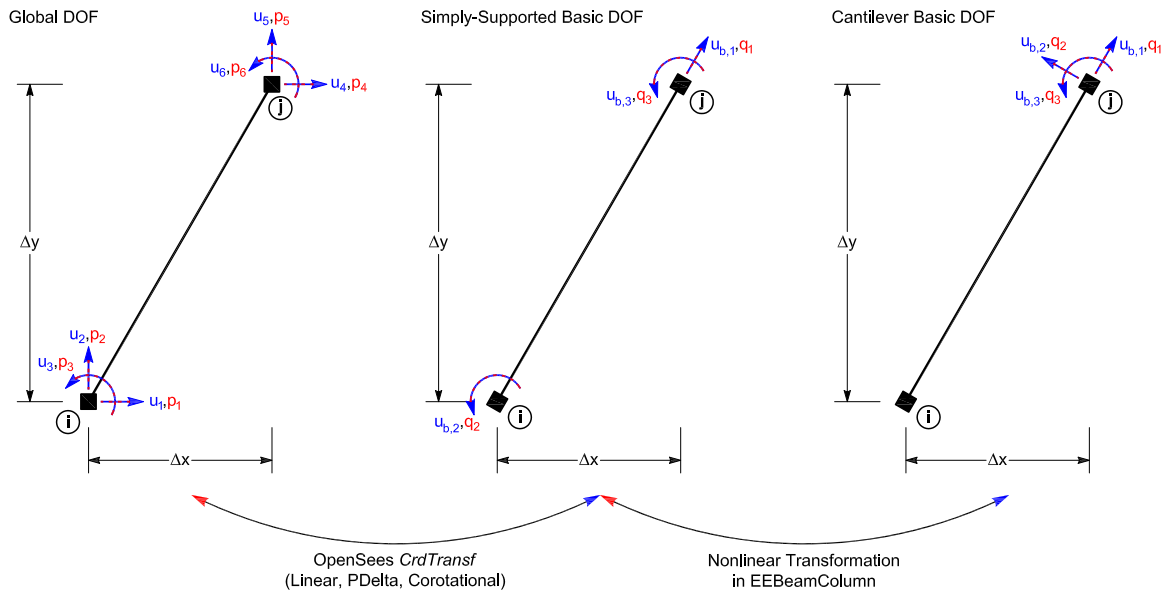


Fig. 3.16 Experimental beam-column element (EEBeamColumn2d).

The experimental elements utilize the OpenSees *CrdTransf* class to transform the response quantities from the global degrees of freedom to the simply supported basic element degrees of freedom and back. The *CrdTransf* class is an abstract base class that has concrete subclasses for linear, p-delta and corotational geometric transformations. This abstraction and encapsulation of the frame element coordinate transformations facilitates the switching among different linear and nonlinear coordinate transformations without affecting the implementations

of the frame elements themselves. Because the simply supported basic system is not well suited for experimental testing (due to the two rotational degrees of freedom), the response quantities are transformed from the simply supported to the cantilever basic system and back. These additional nonlinear transformations are directly implemented in the EEBeamColumn classes.

The nonlinear transformations of the trial displacements from basic system A (simply supported) to basic system B (cantilever) are given by the following expression.

$$\begin{aligned} u_{b,1}^{(B)} &= L_n \cos\left(u_{b,2}^{(A)}\right) - L \qquad with \quad L_n = \left(L + u_{b,1}^{(A)}\right) \\ u_{b,2}^{(B)} &= -L_n \sin\left(u_{b,2}^{(A)}\right) \\ u_{b,3}^{(B)} &= -u_{b,2}^{(A)} + u_{b,3}^{(A)} \end{aligned} \tag{3.1}$$

where L_n is the length of the element in its deformed configuration. The nonlinear transformations for the trial velocities and trial accelerations can be determined by taking derivatives of Equation (3.1) with respect to time. The expressions can easily be evaluated with any computer algebra system (CAS) such as Mathematica (Wolfram) or Maple (Maplesoft), but get quite involved and are therefore not shown here.

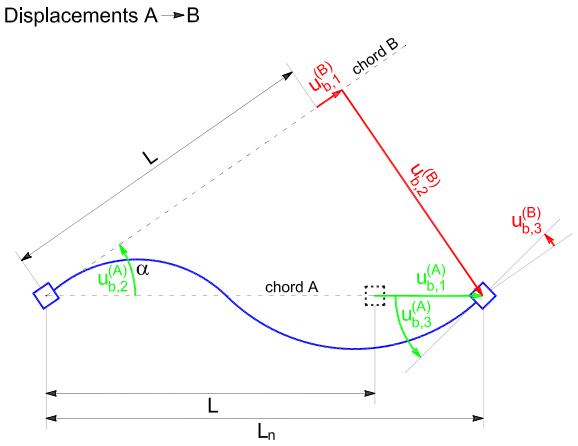


Fig. 3.17 Nonlinear transformation from basic system A to B.

The inverse transformations of measured displacements from basic system B to basic system A take the following form.

$$\begin{aligned} u_{b,1}^{(A)} &= L_n - L \qquad with \quad L_n = \sqrt{\left(L + u_{b,1}^{(B)}\right)^2 + \left(u_{b,2}^{(B)}\right)^2} \\ u_{b,2}^{(A)} &= -\alpha \qquad\qquad\qquad \alpha = \arctan\left(\frac{u_{b,2}^{(B)}}{L + u_{b,1}^{(B)}}\right) \\ u_{b,3}^{(A)} &= -\alpha + u_{b,3}^{(B)} \end{aligned} \tag{3.2}$$

where L_n is the length of the element in its deformed configuration and α is the angle from chord B to chord A. Finally, the measured forces are transformed from basic system B to basic system A as follows.

$$\begin{aligned} q_1^{(A)} &= \cos(\alpha)\, q_1^{(B)} + \sin(\alpha)\, q_2^{(B)} \\ q_2^{(A)} &= u_{b,2}^{(B)}\, q_1^{(B)} - \left(L + u_{b,1}^{(B)}\right) q_2^{(B)} - q_3^{(B)} \\ q_3^{(A)} &= q_3^{(B)} \end{aligned} \tag{3.3}$$

where the chord rotation α is given by the same formula as in (3.2) with the measured displacements replaced by the trial displacements to reduce the effect of experimental errors.

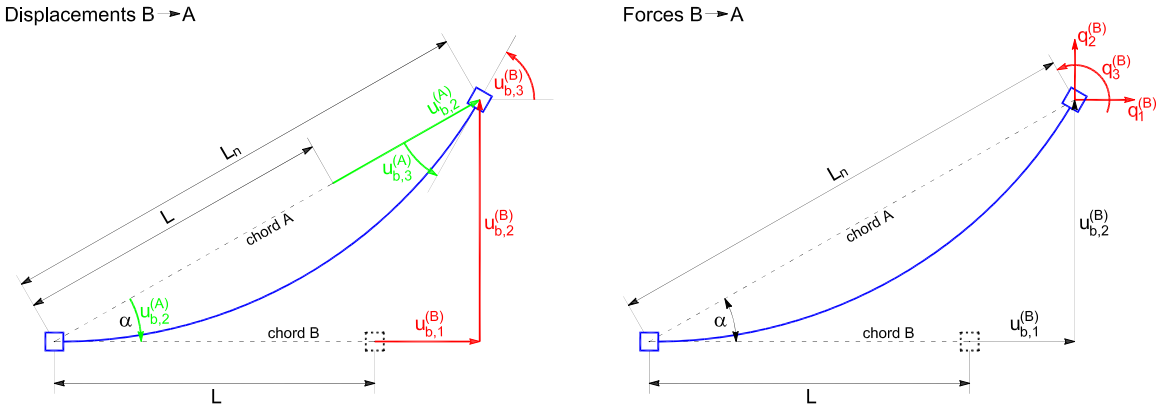


Fig. 3.18 Nonlinear transformations from basic system B to A.

Two-Node Link:

The experimental two-node link element is defined by two nodes and can be used in 1D-, 2D-, and 3D-problems. This element can have 1 to 6 basic degrees of freedom that are uncoupled from each other. It is important to note that the element does not consider geometry as inferred from the given nodal coordinates, but utilizes the local orientation vectors, which are specified by the user, to determine the directions of the springs. The element is ideal for testing plastic hinges and other configurations that can be modeled by a range of uncoupled springs. The transformations of trial displacements and resisting forces from the global degrees of freedom to the 1 to 6 basic degrees of freedom and back are treated as linear geometric transformations and are implemented in the EETwoNodeLink class. As can be seen from Fig. 3.19, the degrees of freedom at which displacements are controlled (blue) and the ones at which resisting forces are acquired (red), are collocated.

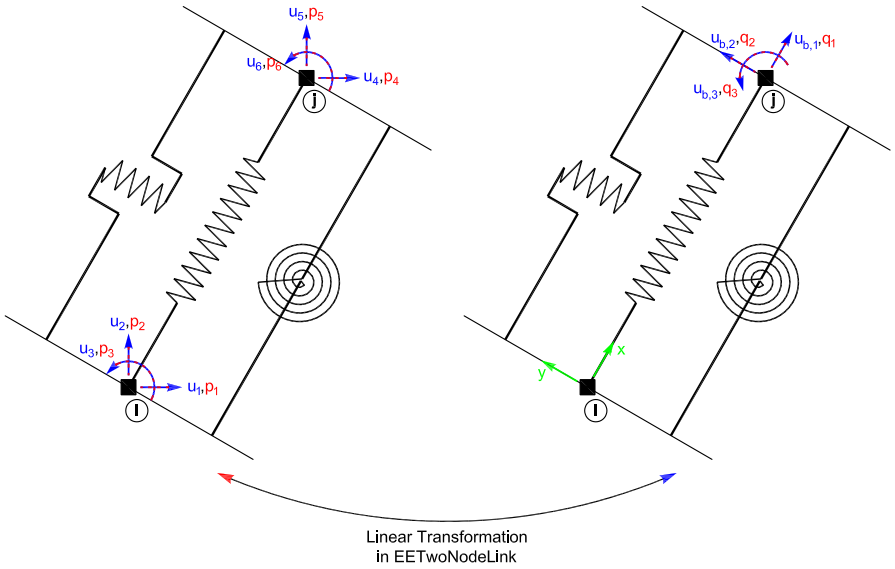


Fig. 3.19 Experimental two-node link element (EETwoNodeLink).

Generic:

The generic experimental element is defined by any number of nodes and any number of the global degrees of freedom at those nodes. The number of degrees of freedom for each node obviously has to be between one and ndf (the maximal number of degrees of freedom of the problem). As such the element can be utilized in 1D-, 2D,- and 3D-problems.

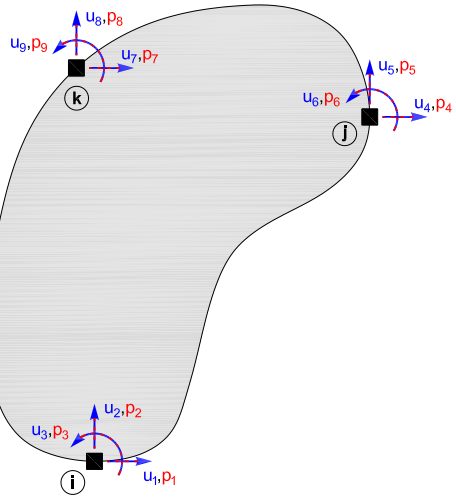


Fig. 3.20 Generic experimental element (EEGeneric).

Since the control and data-acquisition degrees of freedom are already in the global coordinate system, the EEGeneric class does not perform any transformations of the response quantities like the other experimental element subclasses. Instead it extracts the trial response quantities at the appropriate degrees of freedom for control and assembles the measured response

quantities from the acquired signals at the corresponding degrees of freedom. As can be seen from Fig. 3.20, the degrees of freedom at which displacements are controlled (blue) and the ones at which resisting forces are acquired (red), are collocated. This experimental element is useful for representing any general specimen or larger subassemblies of an entire structure for which no specific experimental elements exist, as long as actuators can be attached to such degrees of freedom to control them correctly. Moreover, it can also be employed to represent multiple experimental elements using a single element, as long as the individual elements do not require specialized modeling techniques.

Inverted-V Brace:

The experimental inverted-V brace element is defined by three nodes and can be used in 2D-problems.

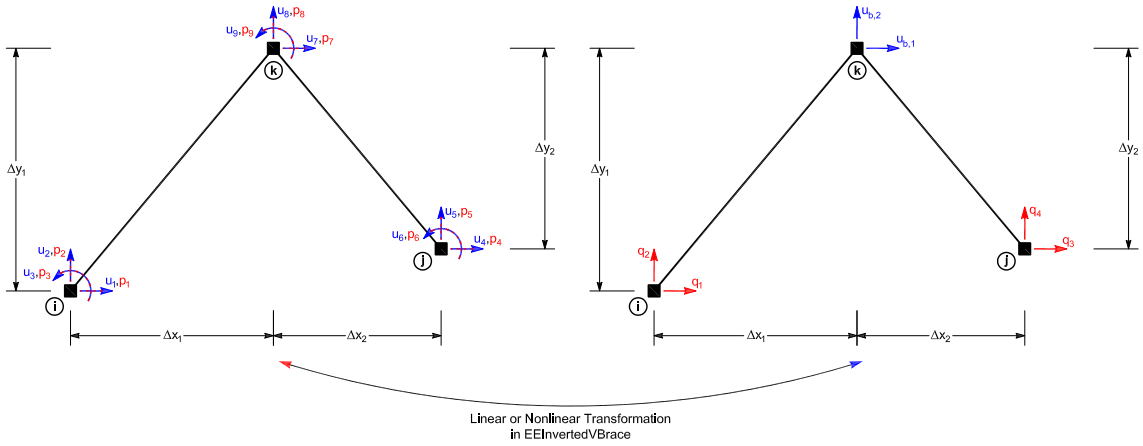


Fig. 3.21 Experimental inverted-V brace element (EEInvertedVBrace2d).

It is assumed that the two braces are pinned at their ends, meaning that nodal rotations are ignored and moments are set to zero. As can be seen from Fig. 3.21 the deformation of the experimental element is controlled by the two translational degrees of freedom at the brace intersection at node k. However, in order to correctly capture the buckling behavior of the inverted-V brace subassembly, resisting forces are acquired by load cells at the two brace supports at nodes i and j, and the forces at node k are then calculated from the element equilibrium. In contrast with the previously discussed experimental elements, the control and data-acquisition degrees of freedom are therefore no longer collocated for the inverted-V brace element. The linear transformations of the trial displacements from the global degrees of freedom to the basic degrees of freedom take the following form.

$$u_b = T\,u \qquad with$$

$$T = \begin{bmatrix} -\frac{\Delta x_1 \Delta y_2}{D} & -\frac{\Delta y_1 \Delta y_2}{D} & 0 & \frac{\Delta y_1 \Delta x_2}{D} & \frac{\Delta y_1 \Delta y_2}{D} & 0 & 1 & 0 & 0 \\ \frac{\Delta x_1 \Delta x_2}{D} & -\frac{\Delta y_1 \Delta x_2}{D} & 0 & -\frac{\Delta x_1 \Delta x_2}{D} & -\frac{\Delta x_1 \Delta y_2}{D} & 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.4}$$

$$D = \Delta x_1 \Delta y_2 - \Delta y_1 \Delta x_2$$

The transformations of the measured forces from the two load cells to the global degrees of freedom account for cross-talk effects between the axial and shear force channels of the load cells. This leads to the following expressions that are implemented in the EEInvertedVBrace2d class.

$$\begin{aligned} p_1 &= \frac{1}{2}\left(q_1 + \frac{\Delta x_1}{\Delta y_1} q_2\right) & p_2 &= \frac{1}{2}\left(\frac{\Delta y_1}{\Delta x_1} q_1 + q_2\right) \\ p_4 &= \frac{1}{2}\left(q_4 + \frac{\Delta x_2}{\Delta y_2} q_5\right) & p_5 &= \frac{1}{2}\left(\frac{\Delta y_2}{\Delta x_2} q_4 + q_5\right) \\ p_7 &= -p_1 - p_4 & p_8 &= -p_2 - p_5 \end{aligned} \tag{3.5}$$

For a more in-depth description of the theory, the nonlinear transformations and several application examples of this experimental element, the reader is referred to (Yang 2006).

3.5.2 Experimental Sites

Local:

As the name suggests, the LocalExpSite class is employed for local testing at a single laboratory. This means that the computational analysis and the experimental testing take place at the same location. Since there are no wide area network communications necessary in such a test, the LocalExpSite class merely passes messages along and stores the trial and output data. Even though the LocalExpSite can therefore be considered a dummy site, it is still required in order to treat local and geographically distributed sites in a uniform manner.

Shadow and Actor:

The ShadowExpSite and ActorExpSite form the client-server architecture that is necessary to geographically distribute computational and experimental sites. The middle-tier server on which the OpenFresco core components are running is therefore further split up into a second and third tier. Since this distribution of processes across a network is very similar to the distributed

computing model in OpenSees, several abstract and concrete classes from such a model are utilized to model the desired behavior.

As can be seen from Fig. 3.22 the ShadowExpSite, which represents the client side, inherits its properties from the *Shadow* base class and the ActorExpSite, which acts as the server, inherits its properties from the *Actor* base class. Both the *Shadow* and the *Actor* parent classes are associated with the abstract *Channel* base class.

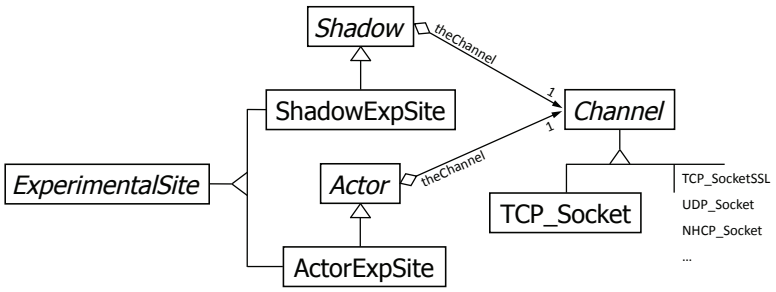


Fig. 3.22 Class diagram of client-server architecture.

Such *Channel* object provides the communication methods between the client and the server processes. The concrete subclasses of the *Channel* class are responsible for implementing the different communication protocols. The TCP_SocketSSL subclass has been added to the existing communication protocols that are available from OpenSees. This new class utilizes the industry-tested, open-source OpenSSL technology to encrypt and decrypt data that are sent across a TCP/IP network. Both the client and the server are being verified during the initial handshake and thus require private keys and security certificates. Hence, the TCP_SocketSSL class provides the means for secure geographically distributed testing, while communication speeds are only slightly reduced by the encryption/decryption processes. In addition, to make geographically distributed testing as efficient as possible, the ShadowExpSite and the ActorExpSite assemble the request identification and all the data into a single message. This significantly reduces the number of required network transactions per computational step, and thus leads to improved performance.

3.5.3 Experimental Setups

No Transformation:

As the name suggests, the ESNoTransformation class is a dummy experimental setup that does not perform any transformations from the basic element degrees of freedom to the actuator

degrees of freedom. However, it allows for reordering degrees of freedom without transforming them. This experimental setup should be employed if the experimental control and data-acquisition systems directly perform all the necessary transformations, meaning that no additional transformations need to be implemented in the experimental setup object.

One Actuator:

The ESOneActuator experimental setup can be utilized to control any one of the basic degrees of freedom of an experimental element. The control and data-acquisition degrees of freedom of the ESOneActuator object are collocated. Since this setup has only one actuator, none of the response quantities are transformed.

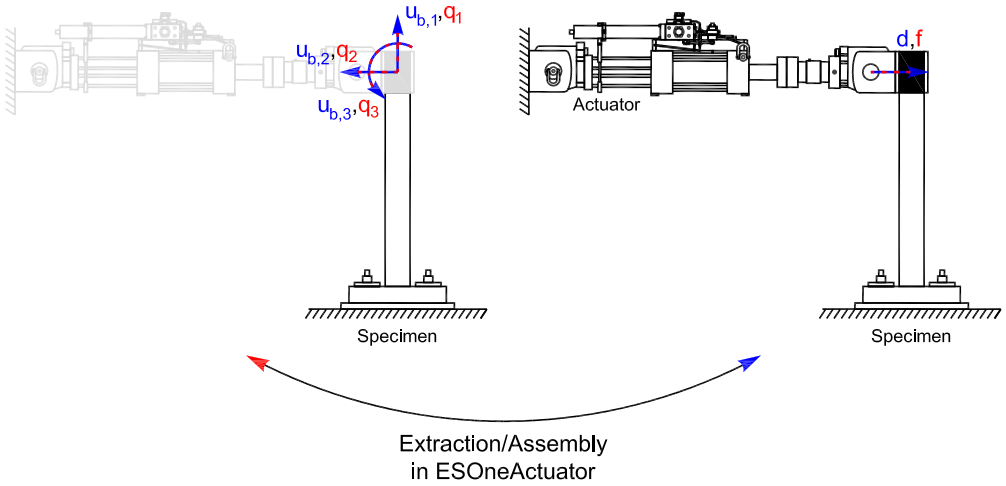


Fig. 3.23 One actuator experimental setup (ESOneActuator).

Instead the ESOneActuator class extracts the trial response quantity at the user-specified degree of freedom for control and assembles the measured response quantity from the acquired signal at the corresponding degree of freedom. Fig. 3.23 shows an experimental beam-column element where the actuator is set to control basic degree of freedom 2 of the element.

Two Actuators:

As can be seen from Fig. 3.24, the ESTwoActuators2d experimental setup configuration consists of two parallel actuators that are connected by a rigid link to control the translational and rotational degrees of freedom of an experimental element.

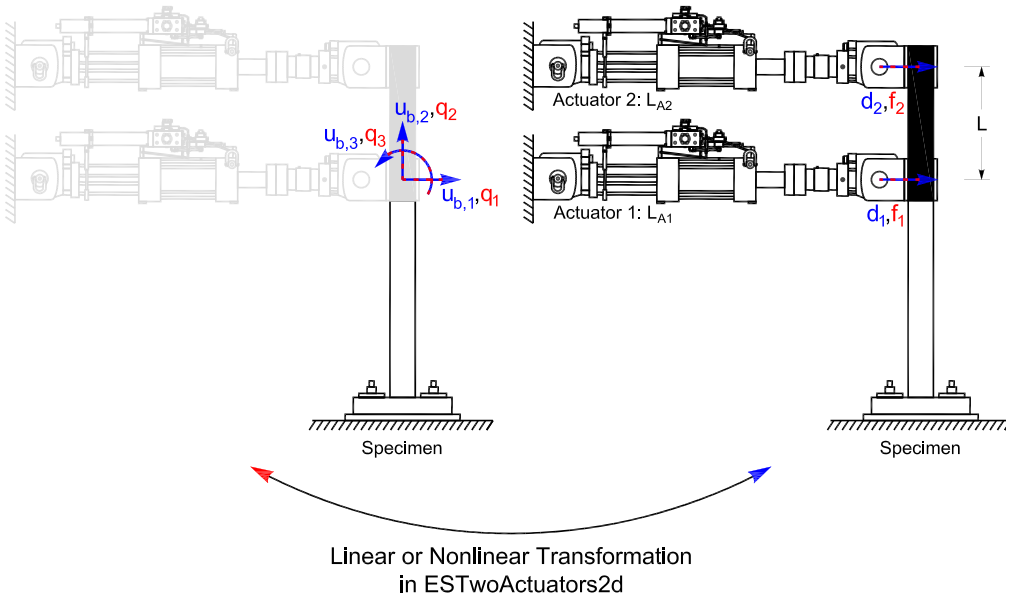


Fig. 3.24 Two actuators experimental setup (ESTwoActuators2d).

The transformations of all the response quantities from the element degrees of freedom to the actuator degrees of freedom and back can be implemented as simple linear geometric transformations or as complex nonlinear geometric transformations that take large displacements into account. To demonstrate the implementation process, the linear as well as nonlinear transformations for the ESTwoActuators2d subclass are summarized next. All the other implementations of the concrete experimental setup subclasses follow a similar approach. The linear (3.6) and nonlinear (3.7) transformations of the trial displacements take the following form.

$$\begin{aligned} d_1 &= u_{b,1} \\ d_2 &= u_{b,1} - L\,u_{b,3} \end{aligned} \tag{3.6}$$

$$\begin{aligned} d_1 &= u_{b,1} \\ d_2 &= \sqrt{\left(u_{b,1} - L\sin\left(u_{b,3}\right) + L_{A1}\right)^2 + \left(L\cos\left(u_{b,3}\right) - L\right)^2} - L_{A1} \end{aligned} \tag{3.7}$$

where L is the length of the rigid link and L_{A1} is the initial length of the first actuator. While the linear transformations for the trial velocities and trial accelerations have the same form as (3.6), the nonlinear ones can be determined by taking the derivatives of Equation (3.7) with respect to time. The expressions can easily be evaluated with any computer algebra system (CAS) such as Mathematica (Wolfram) or Maple (Maplesoft), but get too involved and are therefore not shown here. The linear and nonlinear, inverse transformations of the displacements

measured by the data-acquisition system back to the basic displacements of the experimental element are shown next.

$$\begin{aligned} u_{b,1} &= d_1 \\ u_{b,2} &= 0 \\ u_{b,3} &= \frac{1}{L}\left(d_1 - d_2\right) \end{aligned} \tag{3.8}$$

$$\begin{aligned} u_{b,1} &= d_1 \\ u_{b,2} &= 0 \\ u_{b,3} &= \arctan\left(\frac{L_{A1}+d_1}{L}\right) - \arccos\left(\frac{\left(L_{A1}+d_2\right)^2 - 2L^2 - \left(L_{A1}+d_1\right)^2}{-2L\sqrt{L^2+\left(L_{A1}+d_1\right)^2}}\right) \end{aligned} \tag{3.9}$$

Finally, the linear and nonlinear cases of the force transformations, from the data-acquisition degrees of freedom back to the element basic degrees of freedom, are expressed as follows.

$$\begin{aligned} q_1 &= f_1 + f_2 \\ q_2 &= 0 \\ q_3 &= -L f_2 \end{aligned} \tag{3.10}$$

$$\begin{aligned} q_1 &= f_1 + f_2\cos\left(\theta_2\right) \qquad with\ \ \theta_2 = \arcsin\left(\frac{L\left(1-\cos\left(u_{b,3}\right)\right)}{L_{A1}+d_2}\right) \\ q_2 &= 0 \\ q_3 &= -f_2\cos\left(\theta_2\right)L\cos\left(u_{b,3}\right) - f_2\sin\left(\theta_2\right)L\sin\left(u_{b,3}\right) \end{aligned} \tag{3.11}$$

For the displacements $u_{b,3}$ and d_2, which are necessary for the nonlinear force transformations in (3.11), it is possible to use either the trial values from (3.7) or the measured values from (3.9). However, to reduce the effects of experimental errors it is often advantageous to utilize the trial displacement values.

Three Actuators:

The ESThreeActuators2d experimental setup can be utilized to control the two translational and the rotational degrees of freedom of an experimental element. As can be seen from Fig. 3.25, the control and data-acquisition degrees of freedom of the ESThreeActuators2d object are collocated. The setup consists of a rigid loading beam, which is connected to the specimen, one

horizontal and two vertical, parallel actuators. The three actuators have lengths L_{A1}, L_{A2}, L_{A3} and the two parts of the rigid loading beam have lengths L_1 and L_2. As with the ESTwoActuators2d subclass, the transformations of all the response quantities from the element degrees of freedom to the actuator degrees of freedom and back have been implemented as simple linear geometric transformations and as complex nonlinear geometric transformations that take large displacements into account. However, the formulas are not shown here, since they get quite involved.

The ESThreeActuatorsJntOff2d subclass, which is a very similar experimental setup that accounts for the rigid joint offsets between the actuator clevis hinges and the loading beam, has also been implemented and is available for testing.

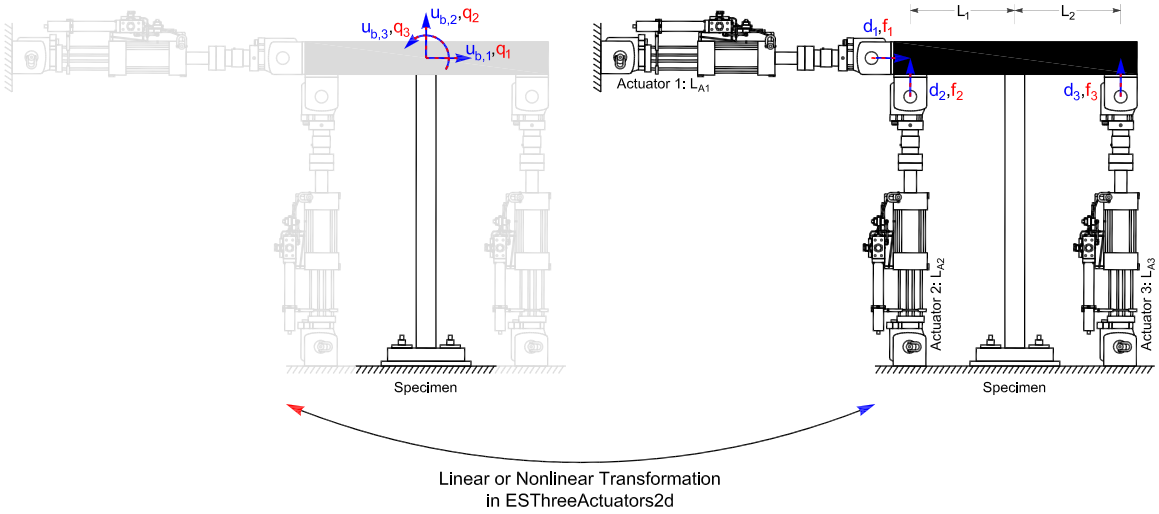


Fig. 3.25 Three actuators experimental setup (ESThreeActuators2d).

Inverted-V Brace:

The ESInvertedVBrace2d experimental setup was specifically developed for the suspended zipper-frame project (Yang 2006). This setup consists of the same loading configuration as the ESThreeActuators2d setup, meaning that a rigid beam and three actuators are utilized to control the two translational and the one rotational degrees of freedom. However, to capture the buckling behavior of the two braces, forces are acquired by two VPM load cells at the brace supports rather than at the actuator degrees of freedom. This means that in contrast to the previously discussed experimental setups, the control and data-acquisition degrees of freedom are no longer collocated here. The three actuators have lengths L_{A1}, L_{A2}, L_{A3} and the two parts of the rigid loading beam have lengths L_1 and L_2. As with the other experimental setup subclass, the

transformations of all the response quantities from the element degrees of freedom to the actuator degrees of freedom and back have been implemented as simple linear geometric transformations and as complex nonlinear geometric transformations that take large displacements into account. For a more in-depth description of such transformations the interested reader is referred to Yang (2006).

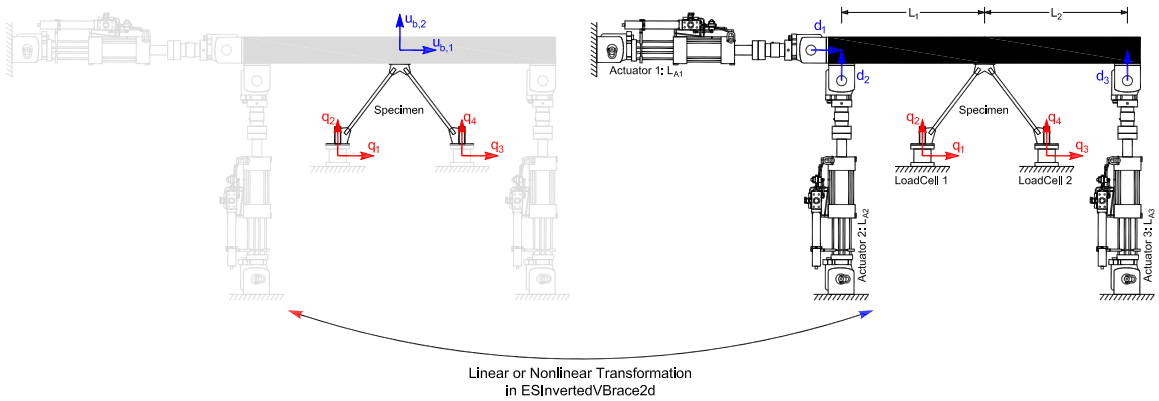


Fig. 3.26 Inverted-V brace experimental setup (ESInvertedVBrace2d).

The ESInvertedVBraceJntOff2d subclass, which is a very similar experimental setup that accounts for the rigid joint offsets between the actuator clevis hinges and the loading beam, has also been implemented and is therefore available for testing.

Aggregator:

As explained before, the ESAggregator subclass is associated with its own *ExperimentalSetup* parent class. This allows two or more concrete experimental setups to be aggregated into a single experimental setup, which then interacts with the *ExperimentalSite* and *ExperimentalControl* objects. In this manner complex experimental testing configurations can be represented by an aggregation of smaller existing experimental setup objects, meaning that no new concrete experimental setup subclasses have to be implemented for testing configurations that are combinations of existing concrete experimental setup subclasses.

3.5.4 Experimental Controls

dSpace, xPC Target and SCRAMNet+:

Even though the dSpace, the xPC Target, and the SCRAMNet+ control objects use different hardware and APIs to interface with the control and data-acquisition systems, they are all based

on the three-loop architecture that is described in detail in Chapter 5. This means that a predictor-corrector algorithm is running on a digital signal processor, which is placed between the machine that is running the analysis and the one that is controlling the actuators.

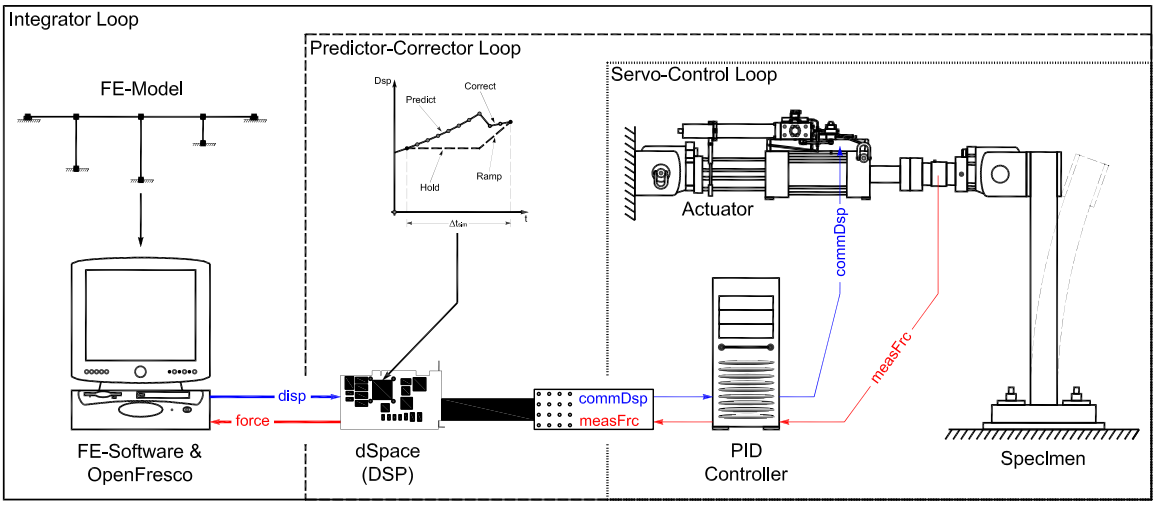


Fig. 3.27 dSpace experimental control (ECdSpace).

The ECdSpace subclass is using the dSpace CLIB API (dSpace Inc.) to access the dSpace digital signal processors. These C libraries provide communication means between a PC and a real-time processor. The API supports the dSpace DS1103 and DS1104 PowerPC real-time processors, which are single PCI hardware boards for PCs. As can be seen from Fig. 3.27, the ECdSpace experimental control object is utilized to communicate with the predictor-corrector algorithm that is executing on the dSpace DSP and generating real-time command signals for the control system. Since the dSpace controller board is directly connected to the analysis machine through a PCI bus, the digital signals (represented by thick lines) are transmitted internally (memory sharing, interrupt lines, etc.). The digital control signals generated by the predictor-corrector algorithm running on the DSP are then converted to analog signals (represented by thin lines) before they are transmitted to the control system.

The xPC Target real-time hardware (Mathworks) is a digital signal processor similar to the dSpace controller board and therefore also provides the functionality to execute the predictor-corrector algorithms in a real-time environment. The concrete ECxPCtarget subclass is using the xPC Target API to interact with an application, which is executing in real time on the target machine. As can be seen from Fig. 3.28, TCP/IP network sockets are used to connect the analysis machine (host) with the digital signal processor (target). On the other hand, the communication

between the DSP and the controller is achieved through SCRAMNet+, an ultra-low-latency, replicated, noncoherent, Shared Common RAM Network. In such a network, data written to memory by a host are sent across fiber optic cables to all the other nodes on the network, meaning that it is almost instantaneously replicated in the memory of all the other machines. In Fig. 3.28 the thick lines again represent digital signals and the thin ones analog.

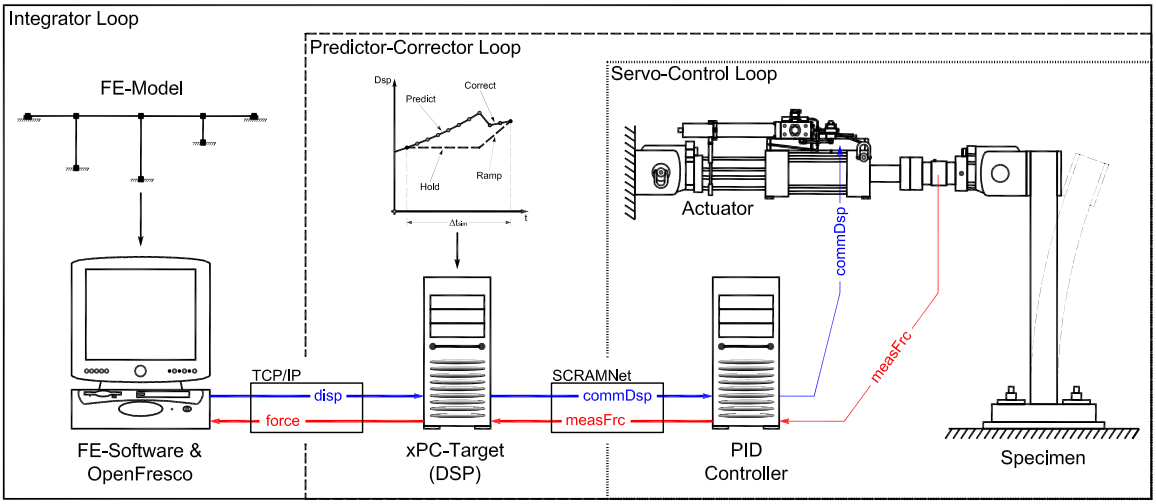


Fig. 3.28 xPC Target experimental control (ECxPCtarget).

For rapid testing with the ECxPCtarget experimental control interface, the TCP/IP connection between the host and the target (which is used by the xPC Target API) might become a bottleneck, thus limiting execution speeds. To circumvent this problem, the concrete ECSCRAMNet experimental control subclass has been implemented.

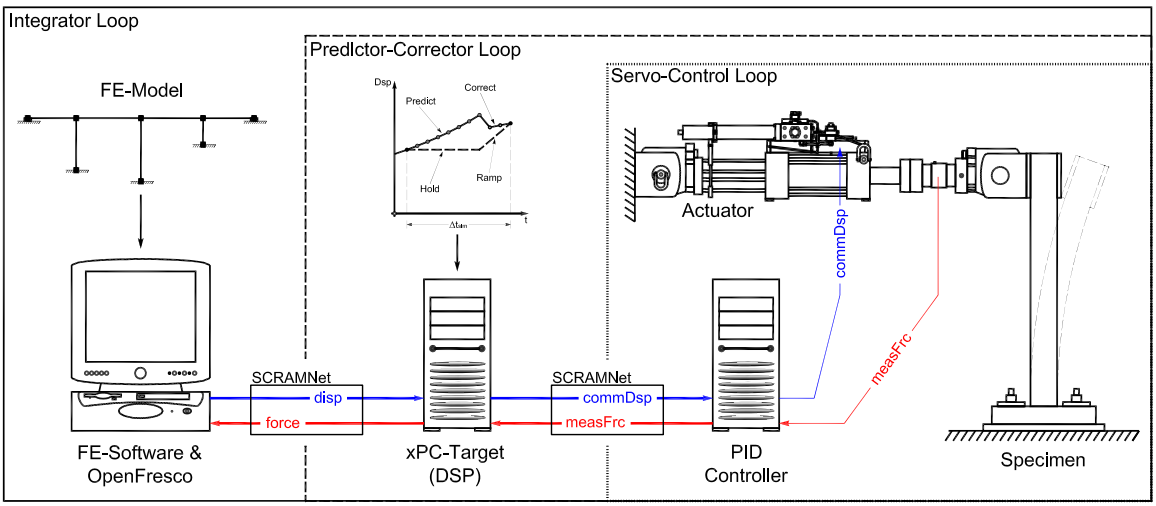


Fig. 3.29 SCRAMNet+ experimental control (ECSCRAMNet).

As can be seen from Fig. 3.29, this interface replaces the slower TCP/IP connection between the analysis machine and the xPC Target digital signal processor with an ultra-low-latency SCRAMNet+ connection (Systran) equivalent to the one between the DSP and the controller. Once again the thick lines represent digital signals and the thin lines represent analog signals.

MTS-CSI and LabVIEW:

The MTS-CSI and LabVIEW experimental control objects are capable of interfacing with MTS hardware (MTS) and the NEES LabVIEW plugin (Hubbard et al. 2004). What these two experimental control objects have in common is that they are both using the abstraction of a control point to define the directions and response quantities that are being controlled and acquired. A control point represents a logical container of one or more directions (degrees of freedom) for sending command signals to a controller or acquiring feedback signals from a data-acquisition system. Thus, control points represent logical groupings of output control channels or input data-acquisition channels (MTS).

The concrete ECMtsCsi subclass is using the newly developed MTS Computer Simulation Interface (CSI) (MTS) to connect to MTS FlexTest controllers with the MTS 793 software. Currently this hardware configuration does not include a predictor-corrector algorithm. This means that continuous testing is not yet possible, since the controller has to fall back to a hold and ramp approach to generate actuator command signals. However, the incorporation of predictor-corrector algorithms within the FlexTest environment has a high priority in the next development cycle.

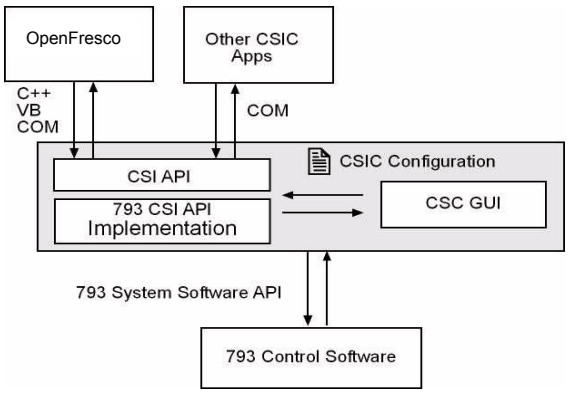


Fig. 3.30 MTS CSI experimental control (ECMtsCsi).

The MTS Computer Simulation Interface and Configurator (CSIC) is an application that provides a high-level programming interface for users to connect their test application to an MTS controller and execute common structural testing commands and data-acquisition operations. Users can take advantage of the simplified graphical user interface (GUI) of the Configurator to create their test application and define control points. Thus, they can focus on the structural analysis rather than learning the detailed, low-level MTS 793 system programming libraries to interface with their MTS controller (MTS).

The ECLabVIEW subclass is employing a single persistent TCP/IP connection to communicate with a LabVIEW controller that is utilizing the NEES LabVIEW NTCP plug-in (Hubbard et al. 2004). The message exchange between the ECLabVIEW experimental control object and the LabVIEW plugin is based on a simple ASCII format that conforms to the LabVIEW plugin specifications. The LabVIEW plugin requires that control commands and data-acquisition requests be expressed in terms of control points. The LabVIEW subclass is therefore associated with the ExperimentalCP class, which defines control points for control and data acquisition (see Fig. 3.6).

Uniaxial Materials and Domain Simulations:

The concrete subclasses of the *ECSimulation* parent class can be used to test the computational model, experimental setup, and network communication to ensure that all non-experimental aspects of a hybrid simulation are functioning properly before conducting an actual experiment.

The ECSimUniaxialMaterials object can be used to simulate a specimen with 1 to n uncoupled degrees of freedom using 1 to n OpenSees uniaxial material models. Since the material models simulate the forces that the 1 to n actuator load cells would measure for a given set of actuator displacements, the units of the specified material values are force and displacement and not stress and strain.

The ECSimDomain subclass makes available in OpenFresco the entire OpenSees domain, which includes the node, element, section, and material libraries. This makes it possible to simulate an actual experimental specimen using OpenSees. The ECSimDomain subclass is associated with the ExperimentalCP class, which defines the control points for control and data acquisition (see Fig. 3.6).

3.6 OPENSEES AS THE COMPUTATIONAL FRAMEWORK

The OpenFresco experimental software framework has been developed so that it can interact with any finite element code that provides an application programming interface (API) and is not a black box. As described in detail in Section 3.3, this flexibility is achieved by employing a multi-tier software architecture that wraps around the OpenFresco core components. However, if OpenSees is utilized as the computational framework, the OpenFresco architecture can significantly be simplified because both software frameworks have the following three things in common: (1) they are both based on the object-oriented software design methodology, (2) they both use the open-source development methodology, and (3) they are both implemented in the C++ programming language. Because the simplified architecture eliminates the network communications between OpenSees (client) and the middle-tier (or application) server, improved performance is attained as compared to the full-blown multi-tier architecture.

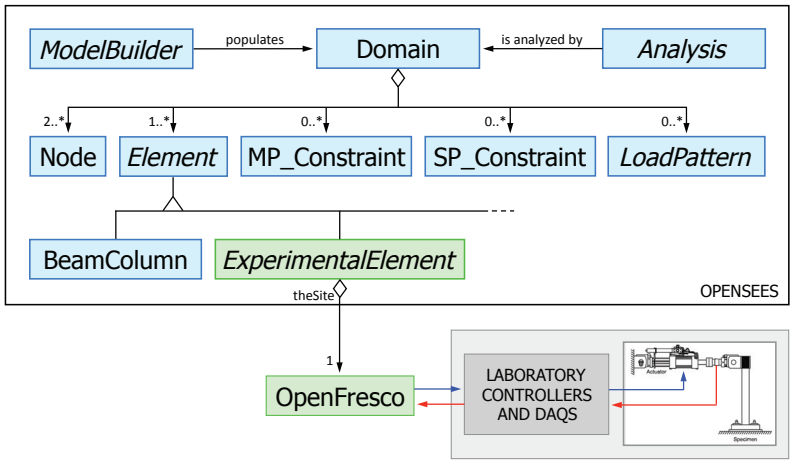


Fig. 3.31 OpenSees as the computational framework.

As can be seen from Fig. 3.31 the abstract *ExperimentalElement* class can directly be embedded into OpenSees, since it is derived from the *Element* base class. The rest of the OpenFresco architecture remains unchanged, meaning that the association between the *ExperimentalElement* class and the *ExperimentalSite* class provides the connection to the other objects. Since in this architecture the experimental elements act as regular elements within OpenSees, they can be used along computational elements without changing OpenSees. At runtime OpenSees dynamically links with OpenFresco and acquires direct access to all the OpenFresco commands without the need for any network communications between the two. The

two software configurations for local and geographically distributed testing, utilizing OpenSees as the computational framework, are shown in Fig. 3.32.

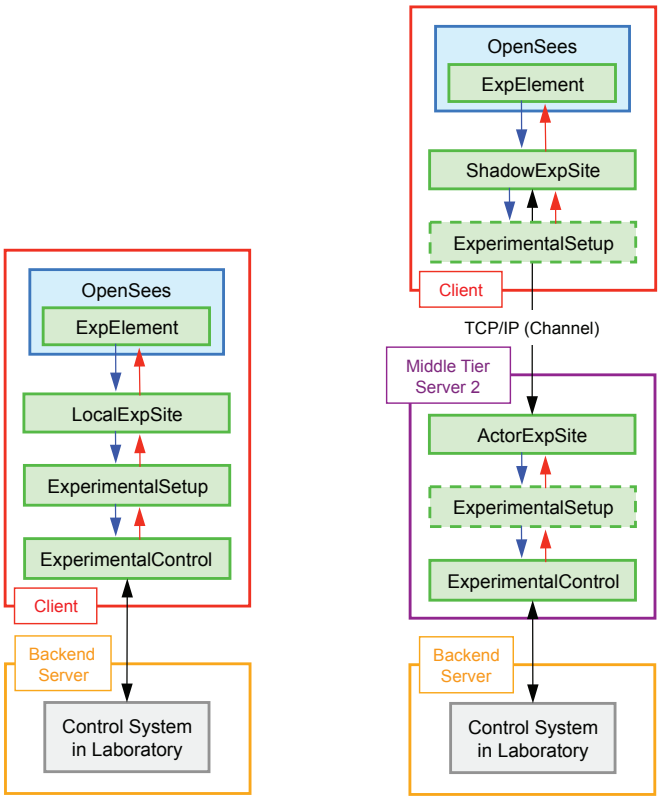


Fig. 3.32 OpenSee-OpenFresco configurations for local and network deployments.

3.7 SUMMARY

The software framework for experimental testing (OpenFresco), which has been presented in this chapter, supports a wide range of computational software, structural testing methods, specimen types, testing configurations, control and data-acquisition systems and communication protocols. The development of such software framework for hybrid simulation is based on the structural analysis approach, in which a hybrid simulation can be viewed as a conventional finite element analysis where physical models of some portions of the structure are embedded in the numerical model. Object-oriented methodologies and a rigorous systems analysis of the operations performed during hybrid simulations were used to determine the fundamental building blocks of the software framework. In order to support any finite element analysis software as the computational driver, a multi-tier client/server architecture was wrapped around those fundamental OpenFresco building blocks. This design approach, which guarantees that the

software framework is environment independent, robust, transparent, scalable, and easily extensible, has been explained in detail. The resulting class diagram, the interfaces for the main base classes and the sequence diagrams have been summarized thereafter. The currently available OpenFresco templates (http://openfresco.neesforge.nees.org), including communication methods and protocols for distributed computation and testing, were discussed next. Finally, the chapter explained how OpenSees can be used as the computational framework and the advantages attained by doing so. The integration methods that are required by the computational drivers to solve the equations of motion and their applicability to hybrid simulation are presented in the next chapter.

4 Integration Methods for Hybrid Simulation

4.1 INTRODUCTION

The linear and nonlinear equations of motion that arise in structural dynamics problems are generally solved through the application of numerical time-stepping integration algorithms. Over the years, a vast number of methods have been developed to numerically solve problems in structural dynamics and other engineering disciplines. However, many of these methods are not particularly well suited to advance the solution during a hybrid simulation test because they were specifically developed for purely analytical problems. Thus, one research area in hybrid simulation has been focusing on the development of specialized direct integration methods that are better suited to obtain reliable solutions when analytical and experimental portions of a structure are mixed and interacting during an analysis.

After a brief introduction to the spatial and time discretization of the structural dynamics problem, this chapter first presents special integration scheme properties required for the reliable, accurate, and fast execution of hybrid simulations. The advantages and disadvantages of explicit, implicit, noniterative, and iterative methods and their application to different problem types are discussed as well. Several existing methods are then summarized including their derivation, accuracy, stability, and implementation. Finally, the chapter presents novel modifications introduced to some of the existing methods as well as the application of several alternative time-stepping integration algorithms, such as the classic Runge-Kutta methods, to hybrid simulation. It is important to notice that the time-stepping integration methods for hybrid simulation are not part of the software framework that has been presented in the previous chapter, but need to be provided by or implemented in the finite element analysis software.

4.2 STRUCTURAL DYNAMICS PROBLEM

4.2.1 Discretization in Space

The derivation of the equations of motion at the discrete degrees of freedom of a structure is initiated by writing out the equilibrium equations for an infinitesimal body, including inertial forces. This leads to the strong form of equilibrium for the structural dynamics problem.

$$\begin{aligned}
&\rho u_{i,tt}(x_i,t) = \sigma_{ji,j}(x_i,t) + b_i(x_i,t) && for\ x_i \in V \\
&u_i(x_i,t) = u_{bi}(x_i,t) && for\ x_i \in \partial V_d \\
&n_j \sigma_{ji}(x_i,t) = t_i(x_i,t) && for\ x_i \in \partial V_t \\
&u_i(x_i,0) = u_{0i}(x_i) \\
&u_{i,t}(x_i,0) = \dot{u}_{0i}(x_i)
\end{aligned} \tag{4.1}$$

where ρ is the mass density of the material, u_i is the displacement vector, σ_{ji} is the stress tensor and b_i are applied body forces per unit volume. Since the differential Equation (4.1) forms a boundary and initial value problem, the remaining four equations define the prescribed displacements u_{bi} and tractions t_i on the boundary of the domain, as well as the initial displacements u_{0i} and initial velocities $\dot{u}_{0i}$ of the problem.

The next step is to construct the weak form of the equilibrium equations, which in mechanics is also called the principle of virtual displacements or virtual energy. To obtain such form the differential Equation (4.1) is first multiplied by a virtual displacement function $\delta u_i(x_i)$, also called variation, and then integrated over the entire domain.

$$\int_V \delta u_i(x_i) \cdot \rho u_{i,tt}(x_i,t)\, dV = \int_V \delta u_i(x_i) \cdot \left(\sigma_{ji,j}(x_i,t) + b_i(x_i,t)\right) dV \tag{4.2}$$

Gauss' divergence theorem is utilized to integrate Equation (4.2) and incorporate the boundary conditions. After simplification and substitution of the boundary value equations, the weak form of the structural dynamics problem is obtained.

$$\begin{aligned}
&\int_V \delta u_i \cdot \rho \ddot{u}_i(t)\, dV + \int_V \delta \varepsilon_{ij} \cdot \sigma_{ji}(t)\, dV = \int_V \delta u_i \cdot b_i(t)\, dV + \oint_{\partial V} \delta u_i \cdot t_i(t)\, dS \\
&\int_V \delta u_i \cdot \rho u_i(0)\, dV = \int_V \delta u_i \cdot \rho u_{0i}\, dV \\
&\int_V \delta u_i \cdot \rho \dot{u}_i(0)\, dV = \int_V \delta u_i \cdot \rho \dot{u}_{0i}\, dV
\end{aligned} \tag{4.3}$$

For simplicity, the displacement field argument x_i on which all functions depend on is suppressed from the equations above. In Equation (4.3) the two left-hand side terms represent the inertial and the resisting forces, which are in equilibrium with the applied body and surface loads on the right-hand side. The remaining two equations correspond to the weak form of the two initial conditions.

In the finite element approach to structural dynamics, the domain of the problem is subsequently discretized in space utilizing elements that are connected by nodes. Thus, the displacement field of the domain needs to be discretized as well, which is achieved by approximating it in terms of element shape functions $\mathbf{N}$ and the discrete displacements at the nodes of the elements.

$$\begin{aligned} &\mathbf{u}(\mathbf{x}) = \mathbf{N}\,\mathbf{u}_{el}\,, \quad \boldsymbol{\varepsilon}(\mathbf{x}) = \mathbf{B}\,\mathbf{u}_{el}\,, \quad \ddot{\mathbf{u}}(\mathbf{x}) = \mathbf{N}\,\ddot{\mathbf{u}}_{el} \\ &\delta\mathbf{u}(\mathbf{x}) = \mathbf{N}\,\delta\mathbf{u}_{el}\,, \quad \delta\boldsymbol{\varepsilon}(\mathbf{x}) = \mathbf{B}\,\delta\mathbf{u}_{el} \end{aligned} \tag{4.4}$$

where $\mathbf{N}$ is a matrix of shape functions and $\mathbf{B}$ is the strain-displacement matrix. The virtual displacement field, which is required in the weak form of the problem, is generally selected to have the same form as the approximate trial displacement field. In the finite element literature, this approach is known as the Galerkin formulation. By combining Equation (4.4) for the trial and virtual displacement fields with the weak form of equilibrium in Equation (4.3) and recognizing that additional concentrated forces and loads can act at the nodes of the structure, the spatially discretized equations of motion are obtained.

$$\begin{aligned} &\mathbf{M}_{\mathbf{nd}}\ddot{\mathbf{U}}(t) + \mathop{\mathbf{A}}_{el}\mathbf{m}_{el}\ddot{\mathbf{u}}_{el}(t) + \mathop{\mathbf{A}}_{el}\mathbf{p}_{\mathbf{r},el}\left(\mathbf{u}_{el}(t), \dot{\mathbf{u}}_{el}(t)\right) = \mathbf{P}(t) - \mathop{\mathbf{A}}_{el}\mathbf{p}_{\mathbf{0},el}(t) \\ &\mathbf{U}(0) = \mathbf{U}_{\mathbf{0}} \\ &\dot{\mathbf{U}}(0) = \dot{\mathbf{U}}_{\mathbf{0}} \end{aligned} \tag{4.5}$$

where the element mass matrices, the element resisting force vectors and the element load vectors are given by the following expressions.

$$\begin{aligned} \mathbf{m}_{el} &= \int_{V_{el}} \mathbf{N}^T \rho \mathbf{N}\; dV \\ \mathbf{p}_{\mathbf{r},el} &= \int_{V_{el}} \mathbf{B}^T \boldsymbol{\sigma}(\boldsymbol{\varepsilon})\; dV \\ \mathbf{p}_{\mathbf{0},el} &= -\int_{V_{el}} \mathbf{N}^T \mathbf{b}\; dV - \int_{\partial V_{t,el}} \mathbf{N}^T \mathbf{t}\; dS - \int_{V_{el}} \mathbf{B}^T \mathbf{D}\boldsymbol{\varepsilon}_{\mathbf{0}}\; dV + \int_{V_{el}} \mathbf{B}^T \boldsymbol{\sigma}_{\mathbf{0}}\; dV \end{aligned} \tag{4.6}$$

In the above Equations (4.5) and (4.6), $\mathbf{M_{nd}}\ddot{\mathbf{U}}$ are the concentrated inertial forces due to the nodal masses, $\mathbf{A}$ is the direct assembly operator, $\mathbf{m}_{el}\ddot{\mathbf{u}}_{el}$ are the element inertial force contributions, $\mathbf{p}_{\mathbf{r},el}$ are the element resisting forces due to internal stresses, $\mathbf{P}$ are the externally applied nodal loads and $\mathbf{p}_{\mathbf{0},el}$ are element forces due to externally applied loads like body forces, boundary tractions, initial strains and initial stresses. Formula (4.5) is also referred to as the semi-discrete equation of motion, since the structural dynamics problem has been discretized in space by the finite element approach but is still continuous in time.

The spatially discretized differential equations can further be simplified by replacing the element assembly operations through global matrices and vectors at the structural degrees of freedom containing all the element contributions. This leads to the general form of the equations of motion at the structural degrees of freedom.

$$\begin{aligned} &\mathbf{M}\,\ddot{\mathbf{U}}(t)+\mathbf{P}_{\mathbf{r}}\left(\mathbf{U}(t),\dot{\mathbf{U}}(t)\right)=\mathbf{P}(t)-\mathbf{P}_{\mathbf{0}}(t) \\ &\mathbf{U}(0)=\mathbf{U}_{\mathbf{0}} \\ &\dot{\mathbf{U}}(0)=\dot{\mathbf{U}}_{\mathbf{0}} \end{aligned} \tag{4.7}$$

where the mass matrix $\mathbf{M}$ is assembled from the nodal and element mass matrices, $\ddot{\mathbf{U}}$ is the acceleration vector at the structural degrees of freedom, $\mathbf{P}_{\mathbf{r}}$ are the assembled element resisting forces (which depend on the structural displacements and velocities), $\mathbf{P}$ are the externally applied nodal loads and $\mathbf{P}_{\mathbf{0}}$ are the assembled element loads. If the applied loads are entirely due to ground accelerations, the nodal load vector $\mathbf{P}$ on the right-hand side of Equation (4.7) can be replaced by the following expression.

$$\mathbf{P}(t)=-\mathbf{M}\,\mathbf{B}\,\ddot{\mathbf{U}}_{g}(t) \tag{4.8}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{B}$ is the ground acceleration transfer matrix and $\ddot{\mathbf{U}}_{g}$ are the specified support accelerations.

4.2.2 Discretization in Time

After the spatial discretization of the differential equations by the finite element method, the system of second-order ordinary differential equations is next discretized in time. An incremental approach is used to solve the equations at successive time steps. Thus, dynamic equilibrium is satisfied at each time step $t_{i+1}=t_i+\Delta t$. The time increments Δt are generally constant, since the applied forces $\mathbf{P}_{i+1}$ obtained from the ground acceleration records are also given at constant time

intervals. This assumption leads to constant time-step integration algorithms in contrast to variable time-step integration algorithms where the step size is adaptively adjusted depending on the current accuracy of the solution.

$$\begin{aligned} &\mathbf{M}\,\ddot{\mathbf{U}}_{i+1} + \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}, \dot{\mathbf{U}}_{i+1}\right) = \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} \\ &\mathbf{U}_{i=0} = \mathbf{U}_{\mathbf{0}} \\ &\dot{\mathbf{U}}_{i=0} = \dot{\mathbf{U}}_{\mathbf{0}} \end{aligned} \tag{4.9}$$

For many problems in structural dynamics, it is oftentimes assumed that the resisting forces $\mathbf{P}_{\mathbf{r}}$ depend linearly on the velocity. For slow hybrid simulations, this is a valid assumption, and the equations of motion can be written in the following form.

$$\begin{aligned} &\mathbf{M}\,\ddot{\mathbf{U}}_{i+1} + \mathbf{C}\,\dot{\mathbf{U}}_{i+1} + \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}\right) = \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} \\ &\mathbf{U}_{i=0} = \mathbf{U}_{\mathbf{0}} \\ &\dot{\mathbf{U}}_{i=0} = \dot{\mathbf{U}}_{\mathbf{0}} \end{aligned} \tag{4.10}$$

where $\mathbf{C}$ is the viscous damping matrix and the resisting forces $\mathbf{P}_{\mathbf{r}}$ are now dependent only on the structural displacements. On the other hand, for real-time hybrid simulations the experimental resisting forces generally contain damping and inertial force contributions from the specimen. Hence, they depend not only on displacements, but also on velocities and accelerations (also see Chapter 2). In this case, the equations of motion take the following form.

$$\begin{aligned} &\mathbf{M}\,\ddot{\mathbf{U}}_{i+1} + \mathbf{C}\,\dot{\mathbf{U}}_{i+1} + \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}, \dot{\mathbf{U}}_{i+1}, \ddot{\mathbf{U}}_{i+1}\right) = \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} \\ &\mathbf{U}_{i=0} = \mathbf{U}_{\mathbf{0}} \\ &\dot{\mathbf{U}}_{i=0} = \dot{\mathbf{U}}_{\mathbf{0}} \end{aligned} \tag{4.11}$$

It should be noticed that in real-time hybrid simulations, the manner in which the calculated command displacements are applied by the control system plays the central role in guaranteeing that consistent velocities and accelerations are obtained from the displacement commands.

Given the numerical solution $\mathbf{U}_i$ at time t_i and satisfying Equation (4.9), (4.10), or (4.11), the task is to advance such solution in time by finding displacement increments $\Delta\mathbf{U}$ such that Equation (4.9), (4.10), or (4.11) is also in equilibrium for $\mathbf{U}_{i+1} = \mathbf{U}_i + \Delta\mathbf{U}$ at time $t_i + \Delta t$. To find these numerical solutions, different strategies can be employed. Most often, the system of second-order ordinary differential equations is directly solved by utilizing integration schemes,

which are based on expressing the displacements $\mathbf{U}_{i+1}$ and velocities $\dot{\mathbf{U}}_{i+1}$ at the new time step $t_i + \Delta t$ in terms of the response quantities at the new time step and k previous time steps.

$$\begin{aligned}\mathbf{U}_{i+1} &= f\left(\mathbf{U}_{i+1}, \dot{\mathbf{U}}_{i+1}, \ddot{\mathbf{U}}_{i+1}, \mathbf{U}_i, \dot{\mathbf{U}}_i, \ddot{\mathbf{U}}_i, \ldots, \mathbf{U}_{i-k+1}, \dot{\mathbf{U}}_{i-k+1}, \ddot{\mathbf{U}}_{i-k+1}\right) \\ \dot{\mathbf{U}}_{i+1} &= f\left(\mathbf{U}_{i+1}, \dot{\mathbf{U}}_{i+1}, \ddot{\mathbf{U}}_{i+1}, \mathbf{U}_i, \dot{\mathbf{U}}_i, \ddot{\mathbf{U}}_i, \ldots, \mathbf{U}_{i-k+1}, \dot{\mathbf{U}}_{i-k+1}, \ddot{\mathbf{U}}_{i-k+1}\right)\end{aligned} \tag{4.12}$$

These schemes are called direct integration methods. For purely analytical structural dynamics problems, the most widely utilized family of solution schemes in the class of direct integration methods is the one developed by Newmark (1959). For this method, the assumed relations among the response quantities (4.12) are finite difference formulas and take the following form.

$$\begin{aligned}\mathbf{U}_{i+1} &= \mathbf{U}_i + \Delta t\, \dot{\mathbf{U}}_i + \frac{\Delta t^2}{2}\left((1-2\beta)\ddot{\mathbf{U}}_i + 2\beta \ddot{\mathbf{U}}_{i+1}\right) \\ \dot{\mathbf{U}}_{i+1} &= \dot{\mathbf{U}}_i + \Delta t\left((1-\gamma)\ddot{\mathbf{U}}_i + \gamma \ddot{\mathbf{U}}_{i+1}\right)\end{aligned} \tag{4.13}$$

where β and γ determine the variation of the accelerations over a time step. It can be seen that the Newmark method is a one-step method, since only responses at $k = 1$ previous time steps are required. However, in order to apply the Newmark method to hybrid simulation problems, several modifications have to be made to the scheme, which are described in Section 4.4.3.

Another strategy for solving the semi-discrete equation of motion in (4.7) is to first convert the second-order differential equation into a system of first-order ordinary differential equations. This transformation is also an essential step in the convergence analysis of direct integration methods and leads to the following system.

$$\begin{aligned}\dot{\mathbf{y}}(t) &= \mathbf{f}(t, \mathbf{y}) = \begin{Bmatrix} \dot{\mathbf{U}}(t) \\ \mathbf{M}^{-1}\left(\mathbf{P}(t) - \mathbf{P_0}(t) - \mathbf{P_r}\left(\mathbf{U}(t), \dot{\mathbf{U}}(t)\right)\right) \end{Bmatrix} \\ \mathbf{y}(0) &= \begin{Bmatrix} \mathbf{U_0} \\ \dot{\mathbf{U}}_\mathbf{0} \end{Bmatrix}\end{aligned} \tag{4.14}$$

where $\mathbf{y} = \left\{\mathbf{U}(t), \dot{\mathbf{U}}(t)\right\}^T$ is a stacked vector containing the structural displacements and velocities. Once the structural dynamics problem is successfully transformed into a first-order initial-value problem, there are a vast number of methods from the field of numerical analysis available to solve such system of first-order differential equations. The easiest method to solve this system is the explicit Euler method that is shown next.

$$\begin{Bmatrix} \mathbf{U} \\ \dot{\mathbf{U}} \end{Bmatrix}_{i+1} = \begin{Bmatrix} \mathbf{U} \\ \dot{\mathbf{U}} \end{Bmatrix}_{i} + \Delta t \begin{Bmatrix} \dot{\mathbf{U}}_i \\ \mathbf{M}^{-1}\left(\mathbf{P}_i - \mathbf{P}_{\mathbf{0},i} - \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_i, \dot{\mathbf{U}}_i\right)\right) \end{Bmatrix}$$
$$\begin{Bmatrix} \mathbf{U} \\ \dot{\mathbf{U}} \end{Bmatrix}_{i=1} = \begin{Bmatrix} \mathbf{U_0} \\ \dot{\mathbf{U}}_\mathbf{0} \end{Bmatrix} \tag{4.15}$$

However, the explicit Euler method is only order $p = 1$ accurate and therefore should not be used for analyses. The most widely used class of such solution schemes is the family of Runge-Kutta methods. Runge-Kutta solution schemes belong to the class of one-step methods, which require only the current solution state at time t_i in order to calculate the new solution at $t_i + \Delta t$. They do not rely on past solution states and are essentially always stable. Multi-step methods, in contrast to one-step methods, form the new solution at time $t_i + \Delta t$ in terms of not only the current solution state but also the k-1 previous solution states. The applicability of Runge-Kutta methods to hybrid simulation is investigated in Section 4.5.

4.3 INTEGRATOR PROPERTIES

During the execution of a hybrid simulation test, the previously derived equations of motion need to be solved similarly to a conventional analytical analysis of a structure. However, since the hybrid model is an aggregation of numerical and experimental portions of a structure, several of the terms that form the equations of motion are assembled from not only analytical elements but also experimental ones. Because experimental elements represent physical specimens in a laboratory, they behave differently than numerical elements and are not always able to execute certain actions that can be performed by analytical elements. This fact leads to a range of special requirements that need to be provided by the integration methods in order to produce reliable and accurate results in a hybrid simulation. In order to effectively discuss these requirements and possible solutions, several general properties of integration schemes are reviewed first. The general properties also provide the basis for grouping the integration methods into different categories.

4.3.1 Explicit vs. Implicit

Besides categorizing integration methods by the number of solution states that are required to advance the analysis in time (one-step, two-step, multi-step), they can also be grouped into

explicit methods versus implicit methods. In this case, the grouping criterion is the new solution's dependence on itself. For an explicit algorithm, the new solution at time $t_i + \Delta t$ can entirely be expressed by known terms such as the current solution state at time t_i and k-1 previous solution states.

$$\mathbf{U}_{i+1} = f\left(\mathbf{U}_i, \dot{\mathbf{U}}_i, \ddot{\mathbf{U}}_i, \ldots, \mathbf{U}_{i-k+1}, \dot{\mathbf{U}}_{i-k+1}, \ddot{\mathbf{U}}_{i-k+1}\right) \tag{4.16}$$

Explicit integration methods are usually conditionally stable, meaning that the time-step size has to be smaller than a critical value to yield a stable solution. This critical value is called the stability limit, which is an important property of an algorithm and depends on the integration method. For explicit methods, the new solution at the end of a time step can often be determined in a single calculation step without the knowledge of the tangent stiffness matrix. The advantages of such methods are that they are computationally very efficient, easy to implement, and fast in their execution. However, because they are conditionally stable they are not well suited for stiff problems and cannot be used for infinitely stiff problems. For structures with very high natural frequencies, the integration time step would have to be so small to satisfy the stability condition, that the application of explicit methods to hybrid simulation becomes impractical.

For an implicit algorithm, the new solution at time $t_i + \Delta t$ not only depends on the known terms at the current and previous time steps, but also on itself. Because of this, implicit algorithms contain algebraic formulas that need to be solved in order to determine the new solution at the end of a time step.

$$\mathbf{U}_{i+1} = f\left(\mathbf{U}_{i+1}, \dot{\mathbf{U}}_{i+1}, \ddot{\mathbf{U}}_{i+1}, \mathbf{U}_i, \dot{\mathbf{U}}_i, \ddot{\mathbf{U}}_i, \ldots, \mathbf{U}_{i-k+1}, \dot{\mathbf{U}}_{i-k+1}, \ddot{\mathbf{U}}_{i-k+1}\right) \tag{4.17}$$

Many implicit integration methods are generally unconditionally stable, making them ideal candidates for stiff and infinitely stiff problems. It also means that only the accuracy of the algorithm needs to be considered when determining the time-step size, since the method is stable for any step size. Usually this permits the selection of larger analysis time steps as compared to explicit methods. Implicit methods are thus better suited for large problems with many degrees of freedom or for infinitely stiff problems, which arise when structural degrees of freedom without mass are present. However, they are computationally more demanding because they require iterative solution schemes, and they can introduce spurious loading cycles on the physical parts of the hybrid model.

4.3.2 Iterative vs. Noniterative

Another important aspect of integration methods is how many function calls they need to make per time step to determine the new solution at $t_i + \Delta t$. For the classic direct integration methods in structural dynamics, a function call is considered to be the determination of the effective forces $\mathbf{P}_{\text{eff}}$ for given displacements, velocities and accelerations.

$$\mathbf{P}_{\text{eff}} = \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} - \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}\right) - \mathbf{C}\,\dot{\mathbf{U}}_{i+1} - \mathbf{M}\,\ddot{\mathbf{U}}_{i+1} \tag{4.18}$$

For Runge-Kutta and multi-step methods, a function call is considered to be the determination of the derivative $\mathbf{f}$ in the first-order ordinary differential equation.

$$\dot{\mathbf{y}} = \mathbf{f} = \begin{Bmatrix} \dot{\mathbf{U}}_{i+1} \\ \mathbf{M}^{-1}\left(\mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} - \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}, \dot{\mathbf{U}}_{i+1}\right)\right) \end{Bmatrix} \tag{4.19}$$

The classic explicit integration methods, which directly solve the second-order differential equations, are noniterative methods because they require only one function call per analysis time step. Explicit Runge-Kutta methods require the same number of function calls as they have stages. Implicit algorithms, on the other hand, contain algebraic formulas that need to be solved in order to determine the new solution at the end of a time step. One approach to solve such implicit equations is to predict a solution using an explicit expression and then subsequently correct it 1 to m-times utilizing the implicit expression. According to the underlying idea, these algorithms are called predictor-multicorrector integration methods. They require a total of m function calls per analysis time step to advance the solution. A second approach to solving the nonlinear implicit equations is to utilize the well-known Newton-Raphson algorithm to find the solution iteratively. The number of function calls is thus directly related to the number of iterations performed per analysis time step. The main disadvantage of implicit methods is that they can be computationally very demanding and far more difficult to implement than explicit methods. For example, in the second approach where the Newton-Raphson algorithm is deployed, each iteration step requires the solution of a system of linear algebraic equations that involves the Jacobian. The formation of the Jacobian and the solution of a large linear system of equations are computationally expensive operations. Algorithms which only update the Jacobian at the beginning of a time step and keep it constant during iterations or that use the initial Jacobian throughout the whole analysis can reduce computational cost. However, they require generally more iterations because quadratic convergence is lost. Another option that can

significantly reduce computational cost is to use a linearly implicit algorithm, which is equivalent to performing one Newton-Raphson iteration only. In the class of Runge-Kutta methods the linearly implicit algorithms are known as Rosenbrock methods. For stiff problems, Rosenbrock methods are the easiest to implement and are thus very popular. The Jacobian that is required for the single Newton-Raphson iteration can again be updated for every time-step or can be determined only once and kept constant throughout the analysis.

4.3.3 Special Requirements

Special requirements, which need to be provided by the integration methods, are discussed in this section. Most of these requirements are essential for an integration method to produce accurate and reliable results in a hybrid simulation.

1. The integration method should be of order $p \geq 2$ accurate. This requirement effectively reduces the accumulation of numerical errors besides the experimental errors, intrinsic in experimental testing methods. The most accurate (smallest error constant) second-order method is the trapezoidal rule, which can be shown to be equivalent with the average acceleration Newmark method.
2. In hybrid simulation, parts of the resisting force vector $\mathbf{P}_r$ are assembled from forces measured in real time in the laboratory. This leads to the requirement that for each integration step the method should make as few function calls as possible, since every function call triggers the acquisition of the resisting forces from the laboratory. The process of acquiring the nonlinear resisting forces from the test structure means that the calculated displacements need to be applied to the specimen by means of a transfer system and the corresponding forces need to be measured with load cells. This process can be time consuming and introduce experimental errors into the numerical integration algorithm.
3. The mass matrix $\mathbf{M}$ in the equations of motion (4.10) is generally singular, since mass moments of inertia at rotational degrees of freedom are often ignored during modeling. This makes the second-order differential equation infinitely stiff. Equations of motion where $\mathbf{M}$ is singular are also called differential algebraic equations and they require unconditionally stable, fully implicit integration methods. In these cases explicit integration methods are either unstable or cannot be applied at all, since the inverse of the

mass matrix $\mathbf{M}$ would be required. For many methods, this requirement conflicts with the previous one, which is to make as few function calls per integration time step as possible.

4. For implicit and linearly implicit integration methods, the parts of the Jacobian that are contributed by the stiffness matrices $\mathbf{k}_{el}$ of the experimental elements are required to remain constant. This is because it is very difficult to obtain an accurate and reliable tangent stiffness matrix of an experimental element from the measurements at the controlled degrees of freedom. Only a few algorithms for estimating a tangent stiffness matrix from experimental measurements have been developed so far, and their application and possible benefits to implicit integration methods have not been demonstrated yet. However, one of these algorithms called the minimal update approach, developed by Igarashi et al. (1993), seems very promising for the problem at hand. The algorithm is based on the BFGS update formula. Its application to implicit integration methods should be further investigated.
5. The integration method should provide some adjustable amount of algorithmic (numerical) energy dissipation to suppress the excitation of higher modes due to experimental errors contaminating the numerical solutions. The propagation of experimental and numerical errors has a more pronounced effect in the higher modes because the cumulative errors increase with the product of the natural frequency and the integration time step (Shing and Mahin 1983). Thus, it is generally viewed as desirable and often considered necessary to have some form of algorithmic damping present to remove the participation of the high-frequency modal components (Hughes 2000). The numerical energy dissipation should damp out the spurious higher-mode participations while maintaining the minimal second-order accuracy of the integration method and not affecting the participation of the lower modes to the solution. Because for second-order integrators the trapezoidal method is the most accurate, any modifications introduced to add algorithmic damping lower the accuracy. Thus, it is important to select integration methods for which the loss of accuracy due to the additional numerical energy dissipation is minimized.
6. Displacement increments calculated by iterative procedures are required to be strictly increasing or strictly decreasing within an integration time step. While this requirement is

not necessary for analytical elements, it is essential for experimental elements that represent physical specimens in a laboratory. Without this restriction, the displacement commands during the iteration process can overshoot the converged displacements. This unintended loading-unloading cycle does not represent the true structural behavior, but is an artifact generated by the numerical algorithm. Contrary to the numerical portions of the hybrid model, for which the response depends on only the committed displacements at convergence, the response of the physical portions is truly path dependent and consequently affected by all iterations.

7. Iterative integration methods should produce displacement increments that are as uniform as possible. This requirement guarantees that the transfer system, which imposes each displacement increment over a typically constant time interval, generates a continuous movement of the test specimen with a uniform speed. It is important to recognize that if the classic Newton-Raphson method is utilized to solve the implicit equations, displacement increments are not uniform but decrease rapidly during iteration. This is due to the quadratic convergence of the algorithm. As long as the control system is allotted the same time interval to impose such displacement increments, this produces velocity and thus force oscillations in the transfer system. These force oscillations feed back into the integration method and can therefore contaminate the solution or in the worst-case cause instability of the algorithm. As such, it is imperative to generate continuous, uniform command signals to avoid force oscillations at all cost.
8. To be real-time compatible an integration scheme needs to execute a constant number of function calls, which produce strictly increasing (respectively decreasing) and uniform displacement increments within each integration time step. This requirement is thus a combination of several of the above requirements.

4.4 DIRECT INTEGRATION METHODS

In direct integration the equations of motion (4.10) are integrated using a numerical step-by-step procedure, the term “direct” meaning that prior to the numerical integration, no transformation of the equations into a different form is carried out (Bathe 1996). The most commonly utilized explicit schemes, including their stability and accuracy, are summarized first. Afterwards, the derivation, implementation, stability, and accuracy of several implicit direct integration methods

are discussed. All schemes are investigated in light of the special hybrid simulation requirements laid out previously.

4.4.1 Explicit Newmark Method (ENM)

The explicit Newmark scheme, which was proposed for hybrid simulation the first time by Mahin and Williams (1980), is the easiest and most straightforward to implement among the class of explicit direct integration methods. It is defined by the temporally discrete equations of motion (4.10) and Newmark's finite difference formulae for displacements and velocities in Equation (4.13). To obtain the explicit form of the algorithm, the finite difference formulae are written in a predictor-corrector form and the β–parameter is set to zero.

$$\begin{aligned}
&\mathbf{M}\,\ddot{\mathbf{U}}_{i+1} + \mathbf{C}\,\dot{\mathbf{U}}_{i+1} + \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}\right) = \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} \\
&\mathbf{U}_{i+1} = \mathbf{U}_i + \Delta t\,\dot{\mathbf{U}}_i + \frac{\Delta t^2}{2}\ddot{\mathbf{U}}_i = \tilde{\mathbf{U}}_{i+1} \\
&\dot{\mathbf{U}}_{i+1} = \dot{\mathbf{U}}_i + \Delta t\left((1-\gamma)\,\ddot{\mathbf{U}}_i + \gamma\,\ddot{\mathbf{U}}_{i+1}\right) = \dot{\tilde{\mathbf{U}}}_{i+1} + \Delta t\,\gamma\,\ddot{\mathbf{U}}_{i+1}
\end{aligned} \tag{4.20}$$

Substitution of the displacement and velocity expressions into (4.20) yields the following system of linear equations, which needs to be solved for the accelerations $\ddot{\mathbf{U}}_{i+1}$ at the new time step $t_i + \Delta t$.

$$\mathbf{M}_{\mathbf{eff}}\,\ddot{\mathbf{U}}_{i+1} = \mathbf{P}_{\mathbf{eff}} \tag{4.21}$$

The effective mass matrix is a combination of the mass and damping matrices with the constants $c_2 = \Delta t\,\gamma$ and $c_3 = 1$. The effective force vector (or unbalanced load vector) is evaluated at the predictor displacements and velocities.

$$\begin{aligned}
\mathbf{M}_{\mathbf{eff}} &= c_3\,\mathbf{M} + c_2\,\mathbf{C} \\
\mathbf{P}_{\mathbf{eff}} &= \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} - \mathbf{P}_{\mathbf{r}}\left(\tilde{\mathbf{U}}_{i+1}\right) - \mathbf{C}\,\dot{\tilde{\mathbf{U}}}_{i+1}
\end{aligned} \tag{4.22}$$

If the effective mass matrix is diagonal, the system of linear equations simplifies to n uncoupled equations that can easily be solved for the new accelerations and that reduce the computational cost significantly. The effective mass matrix becomes diagonal if lumped masses are being used for modeling purposes and no damping or mass-proportional viscous damping is present. Once Equation (4.21) has been solved for the accelerations at $t_i + \Delta t$, the remaining

response quantities such as displacements and velocities are updated. Since the new displacements are equal to the predictor displacements, only the velocities need to be updated.

$$\begin{aligned} \mathbf{U}_{i+1} &= \tilde{\mathbf{U}}_{i+1} \\ \dot{\mathbf{U}}_{i+1} &= \dot{\tilde{\mathbf{U}}}_{i+1} + c_2 \ddot{\mathbf{U}}_{i+1} \end{aligned} \tag{4.23}$$

Repetition of this procedure for the total number of N time steps yields the sought solution. The explicit Newmark integration method can be summarized as pseudo-code as shown in Fig. 4.1.

Initialize:

$\mathbf{U}_0$, $\dot{\mathbf{U}}_0$, $\ddot{\mathbf{U}}_0$

$\mathbf{M}_{\text{eff}} = c_3 \mathbf{M} + c_2 \mathbf{C}$ *(factorized storage)*

Analyze:

for $(i = 0,\ i < N,\ i{+}{+})$

$$\tilde{\mathbf{U}}_{i+1} = \mathbf{U}_i + \Delta t\, \dot{\mathbf{U}}_i + \frac{\Delta t^2}{2} \ddot{\mathbf{U}}_i$$

$$\dot{\tilde{\mathbf{U}}}_{i+1} = \dot{\mathbf{U}}_i + \Delta t (1-\gamma) \ddot{\mathbf{U}}_i$$

$$\mathbf{P}_{\text{eff}} = \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} - \mathbf{P}_{\mathbf{r}}\left(\tilde{\mathbf{U}}_{i+1}\right) - \mathbf{C}\,\dot{\tilde{\mathbf{U}}}_{i+1}$$

$$\ddot{\mathbf{U}}_{i+1} = solve\left(\mathbf{M}_{\text{eff}} \ddot{\mathbf{U}}_{i+1} = \mathbf{P}_{\text{eff}}\right)$$

$$\mathbf{U}_{i+1} = \tilde{\mathbf{U}}_{i+1}$$

$$\dot{\mathbf{U}}_{i+1} = \dot{\tilde{\mathbf{U}}}_{i+1} + c_2 \ddot{\mathbf{U}}_{i+1}$$

end

Fig. 4.1 Explicit Newmark (ENM) method.

It can be shown that the explicit Newmark method with $\gamma = 0.5$ yields the same solutions as the well-known central-difference method. However, even though the two methods are numerically equivalent, the explicit Newmark method has several important advantages over the central-difference method. First, it is a self-starting method, meaning that it does not require any response quantities before $t = 0$. Second, velocities and accelerations are directly obtained as part of the solution algorithm and do not need to be calculated separately. Finally and most important, it has been shown by Shing and Mahin (1983) that the explicit Newmark method possesses more favorable error-propagation characteristics than the central-difference method.

Because of its numerical equivalence, the explicit Newmark method inherits the order of accuracy and the stability condition of the central-difference method. Both methods are second-order accurate $p = 2$. Therefore, they satisfy requirement 1. In addition, the explicit Newmark method is conditionally stable with the following stability limit.

$$\Delta t \leq \frac{(\gamma - 0.5)\xi + \sqrt{0.5\gamma + (\gamma - 0.5)^2 \xi^2}}{\gamma \pi} \quad T_n = \frac{T_n}{\pi} \tag{4.24}$$

where Δt is the step size of the integration method and T_n is the shortest natural period of the structure that is being analyzed. Since the explicit Newmark method is conditionally stable, it cannot be applied to structural problems with singular mass matrices that yield zero-period modes. This violates requirement 3. As can be seen from the pseudo-code in Fig. 4.1, the resisting forces at the predictor displacements are required only once per time step. This means that no more than one force acquisition from the experimental portion of the structure is necessary per integration step. Thus, requirement 2 is satisfied. One of the advantages of the explicit Newmark method is that the stiffness of the structure is not required in the solution algorithm. This is obvious from the effective mass in Equation (4.22), and requirement 4 does therefore not apply. Because of the initial assumption that $\gamma = 0.5$, this integration method is not capable of introducing any numerical damping to suppress higher-mode participation as suggested by requirement 5. On the other hand, for $\gamma > 0.5$ the algorithm introduces too much numerical damping, affecting the participation of the lower modes to the solution. The scheme also is no longer second-order accurate. Finally, requirements 6 and 7 do not apply here, since the explicit Newmark method is not an iterative solution scheme. In summary, except for the lack of algorithmic damping, the explicit Newmark method is an attractive integration scheme for hybrid simulation as long as the conditional stability limit can be satisfied with a reasonable time-step size.

4.4.2 Explicit Generalized-Alpha Method (EGa)

To overcome the aforementioned problem of the explicit Newmark method, many explicit and implicit direct integration methods with more suitable forms of algorithmic energy dissipation have been developed over the years. Among these methods, the generalized-alpha scheme developed by Hulbert and Chung (1996) is the most attractive for hybrid simulation. The method

provides optimized dissipation characteristics of the high-frequency modes, and the amount of algorithmic damping can be specified through the spectral radius ρ_b at the bifurcation frequency $\Omega_b = \omega_b \Delta t$. To introduce the sought numerical energy dissipation, the weighed equations of motion (4.25) are formulated in between time steps t_i and t_{i+1}, depending on the choice of the two weighing parameters α_m and α_f. In addition, to obtain the explicit form of the generalized-alpha method, Newmark's finite difference formulae for displacements and velocities (4.13) are written in a predictor-corrector form as follows.

$$\begin{aligned}
&\mathbf{M}\,\ddot{\mathbf{U}}_{i+\alpha_m} + \mathbf{C}\,\dot{\mathbf{U}}_{i+\alpha_f} + \mathbf{P}_{\mathbf{r},i+\alpha_f} = \mathbf{P}_{i+\alpha_f} - \mathbf{P}_{\mathbf{0},i+\alpha_f} \\
&\mathbf{U}_{i+1} = \mathbf{U}_i + \Delta t\,\dot{\mathbf{U}}_i + \frac{\Delta t^2}{2}(1-2\beta)\,\ddot{\mathbf{U}}_i + \Delta t^2 \beta\,\ddot{\mathbf{U}}_{i+1} = \tilde{\mathbf{U}}_{i+1} + \Delta t^2 \beta\,\ddot{\mathbf{U}}_{i+1} \\
&\dot{\mathbf{U}}_{i+1} = \dot{\mathbf{U}}_i + \Delta t\,(1-\gamma)\,\ddot{\mathbf{U}}_i + \Delta t\,\gamma\,\ddot{\mathbf{U}}_{i+1} = \dot{\tilde{\mathbf{U}}}_{i+1} + \Delta t\,\gamma\,\ddot{\mathbf{U}}_{i+1}
\end{aligned} \tag{4.25}$$

In the above system of balance Equations (4.25), the accelerations and velocities in between time steps are expressed by the following weighed equations.

$$\begin{aligned}
\ddot{\mathbf{U}}_{i+\alpha_m} &= (1-\alpha_m)\,\ddot{\mathbf{U}}_i + \alpha_m\,\ddot{\mathbf{U}}_{i+1} \\
\dot{\mathbf{U}}_{i+\alpha_f} &= (1-\alpha_f)\,\dot{\mathbf{U}}_i + \alpha_f\,\dot{\mathbf{U}}_{i+1}
\end{aligned} \tag{4.26}$$

The internal resisting forces and the externally applied forces, on the other hand, can be expressed through any generalized quadrature rule (Erlicher et al. 2002).

For example, for the generalized trapezoidal rule, the equations take the following form.

$$\begin{aligned}
\mathbf{P}_{\mathbf{r},i+\alpha_f} &= (1-\alpha_f)\,\mathbf{P}_{\mathbf{r}}(\mathbf{U}_i) + \alpha_f\,\mathbf{P}_{\mathbf{r}}(\tilde{\mathbf{U}}_{i+1}) \\
\mathbf{P}_{i+\alpha_f} - \mathbf{P}_{\mathbf{0},i+\alpha_f} &= (1-\alpha_f)(\mathbf{P}_i - \mathbf{P}_{\mathbf{0},i}) + \alpha_f\,(\mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1})
\end{aligned} \tag{4.27}$$

If the generalized midpoint rule is employed instead, the following equations are utilized.

$$\begin{aligned}
\mathbf{P}_{\mathbf{r},i+\alpha_f} &= \mathbf{P}_{\mathbf{r}}(\mathbf{U}_{i+\alpha_f}) = \mathbf{P}_{\mathbf{r}}\left((1-\alpha_f)\,\mathbf{U}_i + \alpha_f\,\tilde{\mathbf{U}}_{i+1}\right) \\
\mathbf{P}_{i+\alpha_f} - \mathbf{P}_{\mathbf{0},i+\alpha_f} &= \mathbf{P}(t_{i+\alpha_f}) - \mathbf{P}_{\mathbf{0}}(t_{i+\alpha_f})
\end{aligned} \tag{4.28}$$

It should be noticed that the predictor displacements, instead of the displacements at the new time step, are being used to evaluate the weighed resisting forces provided by the above generalized trapezoidal and midpoint formulas. Substitution of the displacement and velocity expressions into (4.25) yields the following system of linear equations, which needs to be solved for the accelerations $\ddot{\mathbf{U}}_{i+1}$ at the new time step $t_i + \Delta t$.

$$\mathbf{M}_{\text{eff}}\,\ddot{\mathbf{U}}_{i+1} = \mathbf{P}_{\text{eff}} \tag{4.29}$$

The effective mass matrix is a combination of the mass and damping matrices with the constants $c_1 = \Delta t^2 \beta$, $c_2 = \Delta t\, \gamma$, $c_3 = 1$ and the weighing parameters α_m and α_f. The effective force vector (or unbalanced load vector) also becomes a weighed expression between the current and the predictor response quantities.

$$\begin{aligned}
\mathbf{M}_{\text{eff}} &= \alpha_m c_3\, \mathbf{M} + \alpha_f c_2\, \mathbf{C} \\
\mathbf{P}_{\text{eff}} &= \mathbf{P}_{i+\alpha_f} - \mathbf{P}_{\mathbf{0},i+\alpha_f} - \mathbf{P}_{\mathbf{r},i+\alpha_f} - \mathbf{M}\left(1-\alpha_m\right)\ddot{\mathbf{U}}_i - \mathbf{C}\left[\left(1-\alpha_f\right)\dot{\mathbf{U}}_i + \alpha_f\, \dot{\tilde{\mathbf{U}}}_{i+1}\right]
\end{aligned} \tag{4.30}$$

If the effective mass matrix is diagonal, the system of linear equations simplifies to n uncoupled equations that can easily be solved for the new accelerations and that reduce the computational cost significantly. As with the explicit Newmark method, the effective mass matrix becomes diagonal if lumped masses are being used for modeling purposes and no damping or mass-proportional, viscous damping is present. Once Equation (4.29) has been solved for the accelerations at $t_i + \Delta t$, the remaining response quantities such as displacements and velocities are updated.

$$\begin{aligned}
\mathbf{U}_{i+1} &= \tilde{\mathbf{U}}_{i+1} + c_1\, \ddot{\mathbf{U}}_{i+1} \\
\dot{\mathbf{U}}_{i+1} &= \dot{\tilde{\mathbf{U}}}_{i+1} + c_2\, \ddot{\mathbf{U}}_{i+1}
\end{aligned} \tag{4.31}$$

Repetition of this procedure for the total number of N time steps yields the sought solution. The explicit generalized-alpha integration method can thus be summarized as pseudo-code as shown in Fig. 4.2.

To maximize the high-frequency algorithmic energy dissipation while maintaining second-order accuracy, Hulbert and Chung (1996) derived the following relationships among the algorithmic parameters β and γ, the weighing parameters α_m and α_f, and the spectral radius $0 \le \rho_b \le 1$ at the bifurcation frequency $\Omega_b = \omega_b \Delta t$.

$$\begin{aligned}
\alpha_m &= \frac{2-\rho_b}{1+\rho_b} \quad \in [2.0,\ 0.5] \\
\beta &= \frac{5-3\rho_b-3\alpha_f\left(2+\rho_b-\rho_b^2\right)+\alpha_f^2\left(2+3\rho_b-\rho_b^3\right)}{\left(1-\alpha_f\right)\left(2-\rho_b\right)\left(1+\rho_b\right)^2} \quad \in [2.5,\ 0.0] \\
\gamma &= 0.5+\alpha_m-\alpha_f \quad \in [2.5,\ 0.5]
\end{aligned} \tag{4.32}$$

As can be seen from the above equations, the resulting algorithm is a two-parameter (ρ_b, α_f) scheme that belongs to the family of generalized-alpha methods. If the free weighing parameter $\alpha_f = 0$, the resulting algorithm treats the physical damping explicitly. In contrast, for $\alpha_f \neq 0$, an implicit treatment of the physical damping is obtained. Hulbert and Chung (1996) recommend to use a midpoint evaluation of the physical damping, $\alpha_f = 0.5$, since the method is endowed with a larger stability limit in this neighborhood. The algorithmic energy dissipation of the high-frequency modes increases with the reduction of the spectral radius ρ_b from 1.0 towards 0.0.

Initialize:

$\mathbf{U}_0$, $\dot{\mathbf{U}}_0$, $\ddot{\mathbf{U}}_0$

$\mathbf{M}_{\text{eff}} = \alpha_m c_3 \mathbf{M} + \alpha_f c_2 \mathbf{C}$ *(factorized storage)*

Analyze:

for $(i = 0,\ i < N,\ i{+}{+})$

$$\tilde{\mathbf{U}}_{i+1} = \mathbf{U}_i + \Delta t\, \dot{\mathbf{U}}_i + \frac{\Delta t^2}{2}(1-2\beta)\, \ddot{\mathbf{U}}_i$$

$$\dot{\tilde{\mathbf{U}}}_{i+1} = \dot{\mathbf{U}}_i + \Delta t(1-\gamma)\, \ddot{\mathbf{U}}_i$$

$$\mathbf{P}_{\text{eff}} = \mathbf{P}_{i+\alpha_f} - \mathbf{P}_{\mathbf{0},i+\alpha_f} - \mathbf{P}_{\mathbf{r},i+\alpha_f} \quad \textit{(Eq. 4.27 or Eq. 4.28)}$$
$$-\mathbf{M}(1-\alpha_m)\, \ddot{\mathbf{U}}_i - \mathbf{C}\left[(1-\alpha_f)\, \dot{\mathbf{U}}_i + \alpha_f\, \dot{\tilde{\mathbf{U}}}_{i+1}\right]$$

$$\ddot{\mathbf{U}}_{i+1} = solve\left(\mathbf{M}_{\text{eff}} \ddot{\mathbf{U}}_{i+1} = \mathbf{P}_{\text{eff}}\right)$$

$$\mathbf{U}_{i+1} = \tilde{\mathbf{U}}_{i+1} + c_1\, \ddot{\mathbf{U}}_{i+1}$$

$$\dot{\mathbf{U}}_{i+1} = \dot{\tilde{\mathbf{U}}}_{i+1} + c_2\, \ddot{\mathbf{U}}_{i+1}$$

end

Fig. 4.2 Explicit generalized-alpha (EGα) method.

However, it is important to recognize that with an implicit treatment of physical damping and with spectral radii close to 1.0, the numerical damping can become negative; and thus, introduce energy into the high-frequency modes instead of dissipating energy. This means that in the case of $\alpha_f \neq 0$ the maximal spectral radius ρ_b needs to be limited as a function of the physical damping present.

With the selection of the integration scheme's parameter according to (4.32), both methods are second-order accurate $p = 2$. Therefore, they satisfy requirement 1. For the case

without physical damping, the explicit generalized-alpha method is conditionally stable with the following stability limit.

$$\Delta t \le \frac{1}{2\pi}\sqrt{\frac{12\left(1+\rho_b\right)^3\left(2-\rho_b\right)}{10+15\rho_b-\rho_b^2+\rho_b^3-\rho_b^4}}\ T_n \tag{4.33}$$

On the other hand, if physical damping is present, the stability limit of the algorithm is more stringent than for the undamped case. For the situations where physical damping is treated explicitly, the stability limit is given as follows.

$$\Delta t \le \frac{2\alpha_m\xi-\sqrt{1-4\beta+8\alpha_m\beta+4\alpha_m^2\left(\xi^2-1\right)}}{\left(1+2\alpha_m-4\beta\right)\pi}\ T_n \tag{4.34}$$

In the above two formulas, Δt is the step size of the integration method, and T_n is the shortest natural period of the structure that is being analyzed. Since the explicit generalized-alpha method is conditionally stable for all cases, it cannot be applied to structural problems with singular mass matrices that yield zero-period modes. This violates requirement 3. As can be seen from the pseudo-code in Fig. 4.2 the resisting forces necessary for assembling the effective force vector are required only once per time step. Depending on the employed generalized quadrature rule, the resisting forces are acquired either at the predictor displacements, or at weighed displacements between the current and the predictor displacements. In short, this means that no more than one force acquisition from the experimental portion of the structure is necessary per integration step. Thus requirement 2 is satisfied. As with the explicit Newmark method, the stiffness of the structure is not required in the solution algorithm of the explicit generalized-alpha method. This is obvious from the equation for the effective mass in (4.30), and requirement 4 does therefore not apply. Due to the nature of the method, requirement 5 is satisfied. Specifically, the direct integration scheme provides optimal dissipation of the high-frequency modes, while the amount of algorithmic damping can be specified through the spectral radius ρ_b. In addition, the accuracy of the method is only slightly lowered by the numerical energy dissipation, so that the scheme remains second-order accurate. Finally, requirements 6 and 7 do not apply here, since the explicit generalized-alpha method is not an iterative solution scheme. In summary, as long as the conditional stability limit can be satisfied with a reasonable time-step size, the explicit generalized-alpha method is a very attractive integration scheme for hybrid

simulation. Compared to the explicit Newmark method it has the important advantage of providing an adjustable numerical energy dissipation of the high-frequency modes.

4.4.3 Implicit Newmark Methods (INM1, INM2)

As mentioned earlier the direct integration scheme developed by Newmark (1959) is the most extensively used method for purely analytical structural dynamics problems. However, in order to apply the implicit Newmark method to hybrid simulation problems, the scheme has to be adjusted for a proper interaction with the physical parts of the hybrid model. After the derivation of the algorithm for purely analytical nonlinear problems, several possible modifications of such algorithm and the resulting implicit integration methods applicable to hybrid simulation are discussed.

The implicit Newmark direct integration method is defined by the temporally discrete equations of motion (4.10) and Newmark's finite difference formulae for displacements and velocities (4.13), which are repeated here for convenience.

$$\begin{aligned}
&\mathbf{M}\,\ddot{\mathbf{U}}_{i+1} + \mathbf{C}\,\dot{\mathbf{U}}_{i+1} + \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}\right) = \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} \\
&\mathbf{U}_{i+1} = \mathbf{U}_i + \Delta t\,\dot{\mathbf{U}}_i + \frac{\Delta t^2}{2}\left(\left(1-2\beta\right)\ddot{\mathbf{U}}_i + 2\beta\,\ddot{\mathbf{U}}_{i+1}\right) \\
&\dot{\mathbf{U}}_{i+1} = \dot{\mathbf{U}}_i + \Delta t\left(\left(1-\gamma\right)\ddot{\mathbf{U}}_i + \gamma\,\ddot{\mathbf{U}}_{i+1}\right)
\end{aligned} \tag{4.35}$$

In order to determine the three unknown response quantities, such as displacements, velocities, and accelerations, the three equations in (4.35) can be solved by two different approaches.

D-Form:

The first approach, also called D-form, advances the solution in time by solving for displacement increments. In order to derive such form, Newmark's finite difference formulae (4.35) are first reformulated to obtain expressions for velocities and accelerations at t_{i+1}.

$$\begin{aligned}
\dot{\mathbf{U}}_{i+1} &= \frac{\gamma}{\Delta t\,\beta}\left(\mathbf{U}_{i+1} - \mathbf{U}_i\right) - \left(\frac{\gamma}{\beta} - 1\right)\dot{\mathbf{U}}_i - \Delta t\left(\frac{\gamma}{2\beta} - 1\right)\ddot{\mathbf{U}}_i \\
\ddot{\mathbf{U}}_{i+1} &= \frac{1}{\Delta t^2\beta}\left(\mathbf{U}_{i+1} - \mathbf{U}_i\right) - \frac{1}{\Delta t\,\beta}\dot{\mathbf{U}}_i - \left(\frac{1}{2\beta} - 1\right)\ddot{\mathbf{U}}_i
\end{aligned} \tag{4.36}$$

Substitution of these velocities and accelerations into the balance Equations (4.35) and equating such equations to zero yields a nonlinear system of equations, in which the unknowns are the displacements at the new time step $t_i + \Delta t$.

$$F\left(\mathbf{U}_{i+1}\right) = \mathbf{M}\,\ddot{\mathbf{U}}_{i+1} + \mathbf{C}\,\dot{\mathbf{U}}_{i+1} + \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}\right) - \mathbf{P}_{i+1} + \mathbf{P}_{\mathbf{0},i+1} = \mathbf{0} \tag{4.37}$$

In order to solve such system of nonlinear equations for the unknown displacements the well-known iterative Newton-Raphson algorithm is employed.

$$DF\left(\mathbf{U}_{i+1}^{(k)}\right)\Delta\mathbf{U}^{(k)} = -F\left(\mathbf{U}_{i+1}^{(k)}\right) \tag{4.38}$$

The Jacobian of the vector function DF on the left-hand side and the negative vector function F on the right-hand side both evaluated at the displacements of the current iteration are equivalent to an effective stiffness matrix and an effective force vector (also called unbalanced force vector), respectively.

$$\mathbf{K}_{\mathbf{eff}}^{(k)}\,\Delta\mathbf{U}^{(k)} = \mathbf{P}_{\mathbf{eff}}^{(k)} \tag{4.39}$$

In the above linear system of equations, the effective stiffness matrix and the effective force vector, which depend on the iteration k, take the following form.

$$\begin{aligned}
\mathbf{K}_{\mathbf{eff}}^{(k)} &= c_3\,\mathbf{M} + c_2\,\mathbf{C} + c_1\,\mathbf{K}_{\mathbf{t}}\left(\mathbf{U}_{i+1}^{(k)}\right) \\
\mathbf{P}_{\mathbf{eff}}^{(k)} &= \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} - \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}^{(k)}\right) - \mathbf{M}\,\ddot{\mathbf{U}}_{i+1}^{(k)} - \mathbf{C}\,\dot{\mathbf{U}}_{i+1}^{(k)}
\end{aligned} \tag{4.40}$$

where $c_1 = 1$, $c_2 = \gamma/(\Delta t\,\beta)$, and $c_3 = 1/(\Delta t^2\beta)$. Since the Newton-Raphson algorithm only guarantees convergence if the initial approximation of the displacement vector is sufficiently close to the actual solution, the converged displacement vector at time t_i is used as an initial approximation for the sought displacement vector at time $t_i + \Delta t$.

$$\begin{aligned}
\mathbf{U}_{i+1}^{(k=1)} &= \mathbf{U}_i \\
\dot{\mathbf{U}}_{i+1}^{(k=1)} &= -\left(\frac{\gamma}{\beta} - 1\right)\dot{\mathbf{U}}_i - \Delta t\left(\frac{\gamma}{2\beta} - 1\right)\ddot{\mathbf{U}}_i \\
\ddot{\mathbf{U}}_{i+1}^{(k=1)} &= -\frac{1}{\Delta t\,\beta}\dot{\mathbf{U}}_i - \left(\frac{1}{2\beta} - 1\right)\ddot{\mathbf{U}}_i
\end{aligned} \tag{4.41}$$

Once Equation (4.39) has been solved for the displacement increments at iteration k, the response quantities such as displacements, velocities and accelerations are updated.

$$\begin{aligned}
\mathbf{U}_{i+1}^{(k+1)} &= \mathbf{U}_{i+1}^{(k)} + c_1\,\Delta\mathbf{U}^{(k)} \\
\dot{\mathbf{U}}_{i+1}^{(k+1)} &= \dot{\mathbf{U}}_{i+1}^{(k)} + c_2\,\Delta\mathbf{U}^{(k)} \\
\ddot{\mathbf{U}}_{i+1}^{(k+1)} &= \ddot{\mathbf{U}}_{i+1}^{(k)} + c_3\,\Delta\mathbf{U}^{(k)}
\end{aligned} \tag{4.42}$$

The iterations are repeated until an appropriate convergence criterion is satisfied.

Initialize:

$\mathbf{U}_0,\ \dot{\mathbf{U}}_0,\ \ddot{\mathbf{U}}_0$

Analyze:

for $(i=0,\ i<N,\ i++)$

$$\mathbf{U}_{i+1}^{(k=1)} = \mathbf{U}_i$$

$$\dot{\mathbf{U}}_{i+1}^{(k=1)} = -\left(\frac{\gamma}{\beta}-1\right)\dot{\mathbf{U}}_i - \Delta t\left(\frac{\gamma}{2\beta}-1\right)\ddot{\mathbf{U}}_i$$

$$\ddot{\mathbf{U}}_{i+1}^{(k=1)} = -\frac{1}{\Delta t\,\beta}\dot{\mathbf{U}}_i - \left(\frac{1}{2\beta}-1\right)\ddot{\mathbf{U}}_i$$

do

$$\mathbf{P}_{\mathbf{eff}}^{(k)} = \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} - \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}^{(k)}\right) - \mathbf{M}\,\ddot{\mathbf{U}}_{i+1}^{(k)} - \mathbf{C}\,\dot{\mathbf{U}}_{i+1}^{(k)}$$

$$\mathbf{K}_{\mathbf{eff}}^{(k)} = c_3\,\mathbf{M} + c_2\,\mathbf{C} + c_1\,\mathbf{K}_{\mathbf{t}}\left(\mathbf{U}_{i+1}^{(k)}\right)$$

$$\Delta\mathbf{U}^{(k)} = solve\left(\mathbf{K}_{\mathbf{eff}}^{(k)}\,\Delta\mathbf{U}^{(k)} = \mathbf{P}_{\mathbf{eff}}^{(k)}\right)$$

$$\mathbf{U}_{i+1}^{(k+1)} = \mathbf{U}_{i+1}^{(k)} + c_1\,\Delta\mathbf{U}^{(k)}$$

$$\dot{\mathbf{U}}_{i+1}^{(k+1)} = \dot{\mathbf{U}}_{i+1}^{(k)} + c_2\,\Delta\mathbf{U}^{(k)}$$

$$\ddot{\mathbf{U}}_{i+1}^{(k+1)} = \ddot{\mathbf{U}}_{i+1}^{(k)} + c_3\,\Delta\mathbf{U}^{(k)}$$

while $(criterion > tol;\ k++)$

end

Fig. 4.3 D-form of implicit Newmark method.

Finally, the repetition of this procedure for the total number of N time steps, with Newton-Raphson iterations in each such step, yields the sought solution. The D-form of the implicit Newmark method is summarized as pseudo-code in Fig. 4.3. As can be seen from the pseudo-code, convergence is tested at the end of every iteration step by comparing a convergence criterion against a user-specified tolerance.

Some of the possible criteria that can be employed to measure convergence are explained next. One possible choice is to use the absolute value of the vector norm of the displacement increments. The vector norm can be any p-norm or even the max-norm if the largest component

of the displacement increment matters the most. Another possibility is to utilize relative convergence criteria such as the vector norm of the current displacement increment relative to the vector norm of the first displacement increment or the vector norm of the current displacement increment relative to the sum of all the norms since last convergence. Instead of the displacement increment, it is also possible to use the effective/unbalanced force vector or an energy measure to formulate a convergence criterion, and as before with the displacement increment, these criteria can be expressed in an absolute or relative manner.

Table 4.1 Convergence criteria.

	absolute	*relative*	*relative total*
displacement increment	$\left\|\Delta\mathbf{U}^{(k)}\right\|_p \le tol$	$\frac{\left\|\Delta\mathbf{U}^{(k)}\right\|_p}{\left\|\Delta\mathbf{U}^{(1)}\right\|_p} \le tol$	$\frac{\left\|\Delta\mathbf{U}^{(k)}\right\|_p}{\sum_{j=1}^{k}\left\|\Delta\mathbf{U}^{(j)}\right\|_p} \le tol$
effective/ unbalanced force	$\left\|\mathbf{P}_{\mathbf{eff}}^{(k)}\right\|_p \le tol$	$\frac{\left\|\mathbf{P}_{\mathbf{eff}}^{(k)}\right\|_p}{\left\|\mathbf{P}_{\mathbf{eff}}^{(1)}\right\|_p} \le tol$	-
energy	$\frac{1}{2}\left(\Delta\mathbf{U}^{(k)}\right)^T \mathbf{P}_{\mathbf{eff}}^{(k)} \le tol$	$\frac{\left(\Delta\mathbf{U}^{(k)}\right)^T \mathbf{P}_{\mathbf{eff}}^{(k)}}{\left(\Delta\mathbf{U}^{(1)}\right)^T \mathbf{P}_{\mathbf{eff}}^{(1)}} \le tol$	-

A-Form:

The A-form solution approach is a predictor-multicorrector method, as will be seen shortly. In this form, the direct integration scheme advances the solution in time by solving for acceleration increments instead of displacement increments. To derive the A-form, Newmark's finite difference formulae (4.35) for displacements and velocities are directly substituted into the balance Equations (4.35). Equating everything to zero yields a nonlinear system of equations, in which the unknowns now are the accelerations instead of the displacements at the new time step $t_i + \Delta t$.

$$F\left(\ddot{\mathbf{U}}_{i+1}\right) = \mathbf{M}\,\ddot{\mathbf{U}}_{i+1} + \mathbf{C}\,\dot{\mathbf{U}}_{i+1} + \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}\right) - \mathbf{P}_{i+1} + \mathbf{P}_{\mathbf{0},i+1} = \mathbf{0} \tag{4.43}$$

Once more, the well-known iterative Newton-Raphson algorithm is employed to solve such system of nonlinear equations for the unknown accelerations.

$$DF\left(\ddot{\mathbf{U}}_{i+1}^{(k)}\right)\Delta\ddot{\mathbf{U}}^{(k)} = -F\left(\ddot{\mathbf{U}}_{i+1}^{(k)}\right) \tag{4.44}$$

The Jacobian of the vector function DF on the left-hand side and the negative vector function F on the right-hand side both evaluated at the accelerations of the current iteration are equivalent to an effective mass matrix and an effective force vector, respectively.

$$\mathbf{M}_{\mathbf{eff}}^{(k)}\Delta\ddot{\mathbf{U}}^{(k)} = \mathbf{P}_{\mathbf{eff}}^{(k)} \tag{4.45}$$

In the above linear system of equations, the effective mass matrix and the effective force vector, which depend on the iteration k, take the following form.

$$\begin{aligned}\mathbf{M}_{\mathbf{eff}}^{(k)} &= c_3\,\mathbf{M} + c_2\,\mathbf{C} + c_1\,\mathbf{K}_{\mathbf{t}}\left(\mathbf{U}_{i+1}^{(k)}\right)\\ \mathbf{P}_{\mathbf{eff}}^{(k)} &= \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} - \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}^{(k)}\right) - \mathbf{M}\,\ddot{\mathbf{U}}_{i+1}^{(k)} - \mathbf{C}\,\dot{\mathbf{U}}_{i+1}^{(k)}\end{aligned} \tag{4.46}$$

where $c_1 = \Delta t^2\beta$, $c_2 = \Delta t\,\gamma$ and $c_3 = 1$. Since the Newton-Raphson algorithm only guarantees convergence if the initial approximation of the acceleration vector is sufficiently close to the actual solution, the converged acceleration vector at time t_i is used as an initial approximation for the sought acceleration vector at time $t_i + \Delta t$.

$$\begin{aligned}\mathbf{U}_{i+1}^{(k=1)} &= \mathbf{U}_i + \Delta t\,\dot{\mathbf{U}}_i + \frac{\Delta t^2}{2}\ddot{\mathbf{U}}_i\\ \dot{\mathbf{U}}_{i+1}^{(k=1)} &= \dot{\mathbf{U}}_i + \Delta t\,\ddot{\mathbf{U}}_i\\ \ddot{\mathbf{U}}_{i+1}^{(k=1)} &= \ddot{\mathbf{U}}_i\end{aligned} \tag{4.47}$$

From the structure of Equations (4.47), it becomes obvious why the A-form is a predictor-multicorrector method. In the first iteration of each time step, displacement and velocity predictions are generated, which are then subsequently corrected until convergence is achieved. Once Equation (4.45) has been solved for the acceleration increments at iteration k, the response quantities such as displacements, velocities and accelerations are corrected as follows.

$$\begin{aligned}\mathbf{U}_{i+1}^{(k+1)} &= \mathbf{U}_{i+1}^{(k)} + c_1\,\Delta\ddot{\mathbf{U}}^{(k)}\\ \dot{\mathbf{U}}_{i+1}^{(k+1)} &= \dot{\mathbf{U}}_{i+1}^{(k)} + c_2\,\Delta\ddot{\mathbf{U}}^{(k)}\\ \ddot{\mathbf{U}}_{i+1}^{(k+1)} &= \ddot{\mathbf{U}}_{i+1}^{(k)} + c_3\,\Delta\ddot{\mathbf{U}}^{(k)}\end{aligned} \tag{4.48}$$

The iterations are repeated until an appropriate convergence criterion is satisfied. Finally, the repetition of this procedure for the total number of N time steps, with Newton-Raphson iterations in each such step, yields the sought solution. The A-form of the implicit Newmark method is summarized as pseudo-code in Fig. 4.4.

Initialize:

$\mathbf{U}_0$, $\dot{\mathbf{U}}_0$, $\ddot{\mathbf{U}}_0$

Analyze:

for $(i=0,\ i<N,\ i++)$

$$\mathbf{U}_{i+1}^{(k=1)} = \mathbf{U}_i + \Delta t\,\dot{\mathbf{U}}_i + \frac{\Delta t^2}{2}\ddot{\mathbf{U}}_i$$

$$\dot{\mathbf{U}}_{i+1}^{(k=1)} = \dot{\mathbf{U}}_i + \Delta t\,\ddot{\mathbf{U}}_i$$

$$\ddot{\mathbf{U}}_{i+1}^{(k=1)} = \ddot{\mathbf{U}}_i$$

do

$$\mathbf{P}_{\mathbf{eff}}^{(k)} = \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} - \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}^{(k)}\right) - \mathbf{M}\,\ddot{\mathbf{U}}_{i+1}^{(k)} - \mathbf{C}\,\dot{\mathbf{U}}_{i+1}^{(k)}$$

$$\mathbf{M}_{\mathbf{eff}}^{(k)} = c_3\,\mathbf{M} + c_2\,\mathbf{C} + c_1\,\mathbf{K}_{\mathbf{t}}\left(\mathbf{U}_{i+1}^{(k)}\right)$$

$$\Delta\ddot{\mathbf{U}}^{(k)} = solve\left(\mathbf{M}_{\mathbf{eff}}^{(k)}\,\Delta\ddot{\mathbf{U}}^{(k)} = \mathbf{P}_{\mathbf{eff}}^{(k)}\right)$$

$$\mathbf{U}_{i+1}^{(k+1)} = \mathbf{U}_{i+1}^{(k)} + c_1\,\Delta\ddot{\mathbf{U}}^{(k)}$$

$$\dot{\mathbf{U}}_{i+1}^{(k+1)} = \dot{\mathbf{U}}_{i+1}^{(k)} + c_2\,\Delta\ddot{\mathbf{U}}^{(k)}$$

$$\ddot{\mathbf{U}}_{i+1}^{(k+1)} = \ddot{\mathbf{U}}_{i+1}^{(k)} + c_3\,\Delta\ddot{\mathbf{U}}^{(k)}$$

while $(criterion > tol;\ k++)$

end

Fig. 4.4 A-form of implicit Newmark method.

Similarly to the D-form of the solution scheme, convergence is tested at the end of every iteration step by comparing a convergence criterion against a user-specified tolerance. The possible criteria that can be employed to measure convergence are, however, somewhat different than in the D-form implementation. For the first category of criteria, where convergence is tested on the incremental solution vector, acceleration increments are utilized instead of displacement increments. The second category remains unchanged, meaning that convergence is measured by the absolute or relative size of the effective/unbalanced force vector. Finally, the last category does not apply to the A-form of the solution approach because it does not represent a meaningful physical quantity, such as energy.

While both the D-form and the A-form implementations of the implicit Newmark method are theoretically sound solutions, experience has shown that the D-form implementation encounters fewer convergence issues than the A-form implementation. This is particularly true for complex structural models with strong material and geometric nonlinearities. Because of

these possible convergence issues and other reasons explained next, the D-form implementation of the implicit Newmark method is better suited to be modified for hybrid simulation.

In their current form, both of the presented solution approaches for the implicit Newmark method require a tangent stiffness matrix and produce nonuniform, rapidly-decreasing displacement increments, due to the quadratic convergence of the Newton-Raphson algorithm during iteration. Thus, to obtain suitable forms of the implicit Newmark integration method, several modifications need to be made to the solution schemes. Some of these possible approaches are explained next.

Increment Reduction Factor Modification (INM1):

Because it is difficult to obtain an accurate and reliable tangent stiffness matrix of an experimental element from the measurements at the controlled degrees of freedom, the parts of the Jacobian in (4.40) and (4.46) that are contributed by the tangent stiffness matrices $\mathbf{k}_{\mathbf{t},el}$ of the experimental elements are replaced by initial stiffness matrices $\mathbf{k}_{\mathbf{i},el}$. In addition, to obtain more uniform convergence for certain problem types and structural models, it is sometimes advantageous to utilize initial stiffness matrices for not only the experimental elements but also the analytical ones. However, a mixed treatment, where a structure is divided into partitions that utilize tangent element stiffness matrices and partitions that use initial element stiffness matrices, generally achieves improved convergence. This approach is comparable to implicit-explicit methods, where a structure is divided into partitions that treat elements implicitly and partitions that treat elements explicitly. The selection of appropriate convergence criteria guarantees that all partitions, also the slower converging initial stiffness matrix partitions, reach satisfactory equilibrium. In the first case, this leads to a modified Newton-Raphson algorithm that is using a constant initial stiffness matrix, and thus, a constant Jacobian, which needs to be factorized only once. In the second case, the global stiffness matrix is a nonconstant, mixed matrix, assembled from tangent and initial stiffness element matrices. Thus, the Jacobian is nonconstant as well. While for both cases, quadratic convergence is reduced to linear convergence, the reduction is obviously more pronounced for the constant initial stiffness matrix iteration approach. Nevertheless, the benefit of a single matrix factorization is generally outweighed by the excessive number of iterations required to reach equilibrium.

To improve the reduced order of convergence, Accelerated-Newton methods can be employed to find equilibrium. Accelerated-Newton algorithms hold the stiffness matrix constant

during iteration but use selected information from previous iterations to accelerate convergence. The Krylov-Subspace Accelerated-Newton algorithm developed for nonlinear structural problems by Scott and Fenves (2003) is one of such methods. This Krylov-Newton method has been implemented in OpenSees and can thus easily be utilized for hybrid simulation.

The described modifications of the equilibrium solution algorithms do however not yet guarantee that displacement increments calculated by these iterative procedures are strictly increasing or strictly decreasing within an integration time step, as set forth by requirement 6 for hybrid simulation. As suggested by Shing and Vannan (1991) a reduction factor, θ, that modifies the response increments, can be introduced to enhance the smoothness of the convergence path and to reduce the possibility of spurious loading/unloading cycles during iteration. It is obvious that this unfortunately lowers the order of convergence even further. In addition, the response reduction factor, as described by Shing and Vannan (1991), can be incorporated into only the D-form of the implicit Newmark method. For the A-form, the reduction factor does not achieve the desired effect, since the predictor quantities remain unmodified and the initial displacement increment is therefore not being reduced.

Thus, a first implicit Newmark method for hybrid simulation is obtained from the D-form of the algorithm by utilizing a modified stiffness matrix (initial or mixed) and the increment reduction factor in the iterative solution process. The implicit Newmark method for hybrid simulation is summarized as pseudo-code in Fig. 4.5.

The accuracy and stability of the method depend on the two algorithmic parameters, β and γ. As with the explicit counterpart, the algorithm is only second-order accurate if $\gamma = 0.5$. For $\gamma > 0.5$ the integration scheme is reduced to first-order accuracy. In addition, it introduces too much algorithmic damping, affecting the participation of the lower modes to the solution. The stability limit of the implicit Newmark direct integration method is defined by the following expression.

$$\Delta t \le \frac{(\gamma - 0.5)\xi + \sqrt{0.5\gamma - \beta + (\gamma - 0.5)^2 \xi^2}}{(\gamma - 2\beta)\pi} T_n \tag{4.49}$$

where Δt is the step size of the integration method and T_n is the shortest natural period of the structure that is being analyzed. As can be seen from the denominator in (4.49), the integration method becomes unconditionally stable if the algorithmic parameters $\gamma = 0.5$ (for

second-order accuracy) and $\beta \geq 0.25$ are selected. The implicit Newmark integrator with $\beta = 0.25$ is also known as the average acceleration, or the trapezoidal method.

Initialize:

$\mathbf{U}_0$, $\dot{\mathbf{U}}_0$, $\ddot{\mathbf{U}}_0$

$\mathbf{K}_{\mathbf{eff}} = c_3 \mathbf{M} + c_2 \mathbf{C} + c_1 \mathbf{K}_{\mathbf{i}}$ *(case 1 : initial stiffness)*

Analyze:

for $(i = 0,\ i < N,\ i{+}{+})$

$$\mathbf{U}_{i+1}^{(k=1)} = \mathbf{U}_i$$

$$\dot{\mathbf{U}}_{i+1}^{(k=1)} = -\left(\frac{\gamma}{\beta} - 1\right)\dot{\mathbf{U}}_i - \Delta t\left(\frac{\gamma}{2\beta} - 1\right)\ddot{\mathbf{U}}_i$$

$$\ddot{\mathbf{U}}_{i+1}^{(k=1)} = -\frac{1}{\Delta t\,\beta}\dot{\mathbf{U}}_i - \left(\frac{1}{2\beta} - 1\right)\ddot{\mathbf{U}}_i$$

$$\mathbf{P}_{\mathbf{eff}}^{(k=1)} = \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} - \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}^{(k=1)}\right) - \mathbf{M}\,\ddot{\mathbf{U}}_{i+1}^{(k=1)} - \mathbf{C}\,\dot{\mathbf{U}}_{i+1}^{(k=1)}$$

do

$$\mathbf{K}_{\mathbf{eff}}^{(k)} = c_3 \mathbf{M} + c_2 \mathbf{C} + c_1 \mathbf{K}_{\mathbf{m}}\left(\mathbf{U}_{i+1}^{(k)}\right) \quad \textit{(case 2 : mixed stiffness)}$$

$$\Delta\mathbf{U}^{(k)} = solve\left(\mathbf{K}_{\mathbf{eff}}^{(k)}\,\Delta\mathbf{U}^{(k)} = \mathbf{P}_{\mathbf{eff}}^{(k)}\right)$$

$$\mathbf{U}_{i+1}^{(k+1)} = \mathbf{U}_{i+1}^{(k)} + c_1\,\theta\,\Delta\mathbf{U}^{(k)}$$

$$\dot{\mathbf{U}}_{i+1}^{(k+1)} = \dot{\mathbf{U}}_{i+1}^{(k)} + c_2\,\theta\,\Delta\mathbf{U}^{(k)}$$

$$\ddot{\mathbf{U}}_{i+1}^{(k+1)} = \ddot{\mathbf{U}}_{i+1}^{(k)} + c_3\,\theta\,\Delta\mathbf{U}^{(k)}$$

$$\mathbf{P}_{\mathbf{eff}}^{(k+1)} = \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} - \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}^{(k+1)}\right) - \mathbf{M}\,\ddot{\mathbf{U}}_{i+1}^{(k+1)} - \mathbf{C}\,\dot{\mathbf{U}}_{i+1}^{(k+1)}$$

while $(criterion > tol;\ k{+}{+})$

end

Fig. 4.5 First modified D-form of implicit Newmark (INM1) method.

To investigate the progression of convergence during the equilibrium solution process, two hybrid simulation models are analyzed by the modified implicit Newmark method. The first model is a four-story frame with linear elastic columns that are pinned at the base. The first- and second-story beams are represented by nonlinear experimental beam-column elements and the third- and fourth-story beams are modeled as nonlinear force-beam-column elements with fiber sections at five integration points. Since the structure is symmetric and the loading is asymmetric, only the left half of the structure is modeled for analysis. Only material nonlinearities are considered in this model, meaning that geometric nonlinearities due to axial

forces are ignored. The second model is a portal frame (similar to the one in Chapter 6) defined by two nonlinear columns that are connected by a linear-elastic beam. The left column is represented by a nonlinear experimental beam-column element and the right column is modeled as a nonlinear force-beam-column element with fiber sections at five integration points. Significant gravity loads corresponding to the nodal masses are applied at the top of the two columns, and the resulting large P-Δ effects are included in the modeling of the two column elements. Thus, both the nonlinear geometry and the nonlinear material behavior of the portal-frame columns are taken into account during the hybrid simulation. Both models were analyzed by applying initial stiffness iterations, mixed stiffness iterations, no increment reduction, increment reduction, Newton-Raphson iterations, and Krylov-Newton iterations. The two-norm of the current displacement increment relative to the sum of all the two-norms from last convergence was used as the convergence criterion for all the analyses.

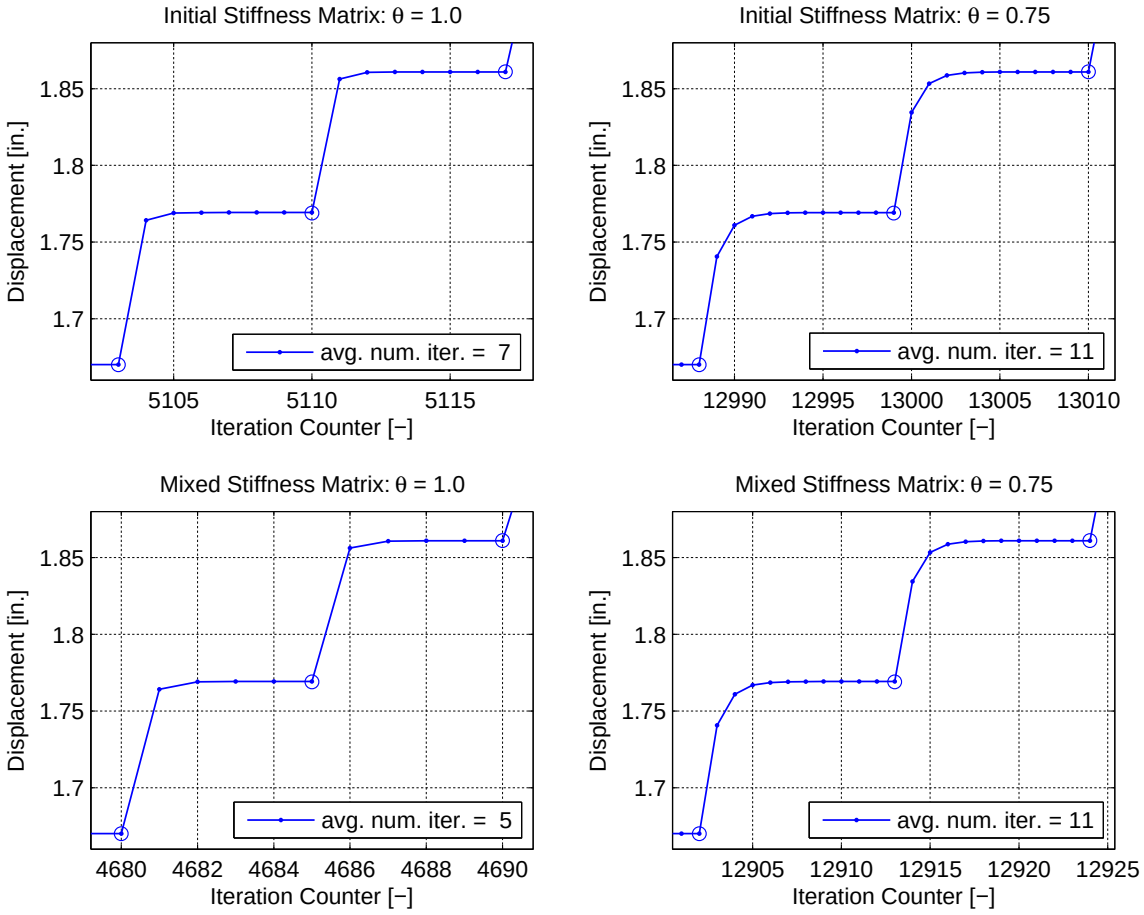


Fig. 4.6 Newton-Raphson convergence comparison for four-story-frame model.

Figures 4.6–4.9 show close-ups of the displacement convergence progress at the controlled degree of freedom of the experimental element over two integration time steps for different analysis parameters. As can be seen from the plots for the four-story-frame model with the Newton-Raphson equilibrium solution algorithm (Fig. 4.6), the application of the increment reduction factor successfully enhances the smoothness of the convergence path and reduces the possibility of spurious loading/unloading cycles during iteration. On the other hand, the application of the mixed stiffness matrix versus the initial stiffness matrix has a negligible effect on the convergence path of this specific model. Similar results are obtained for the Krylov-Newton equilibrium solution algorithm (Fig. 4.7). However, the increment reduction factor is less effective, and the use of the mixed stiffness matrix achieves slightly faster convergence.

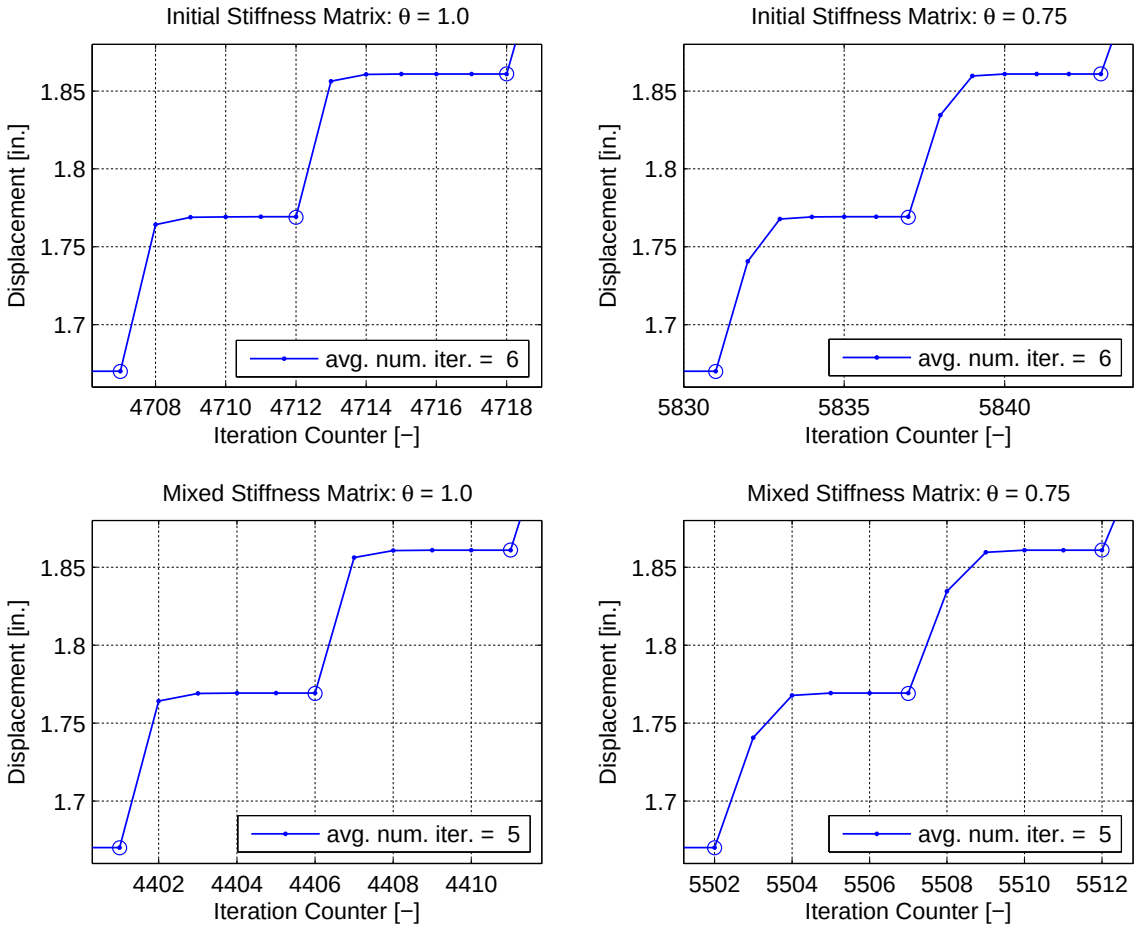


Fig. 4.7 Krylov-Newton convergence comparison for four-story-frame model.

The best convergence path for the four-story-frame model is achieved by employing the Krylov-Newton algorithm with the mixed stiffness matrix and with a moderate increment reduction factor.

The portal-frame model, with its strong geometric and material nonlinearities, exhibits drastically different convergence paths. For the Newton-Raphson algorithm with initial stiffness iterations (Fig. 4.8), the displacement increments are not strictly increasing or strictly decreasing within an integration time step, and thus, introduce spurious loading/unloading cycles during equilibrium iterations. Even an increment reduction factor of 50% cannot entirely resolve this undesirable behavior.

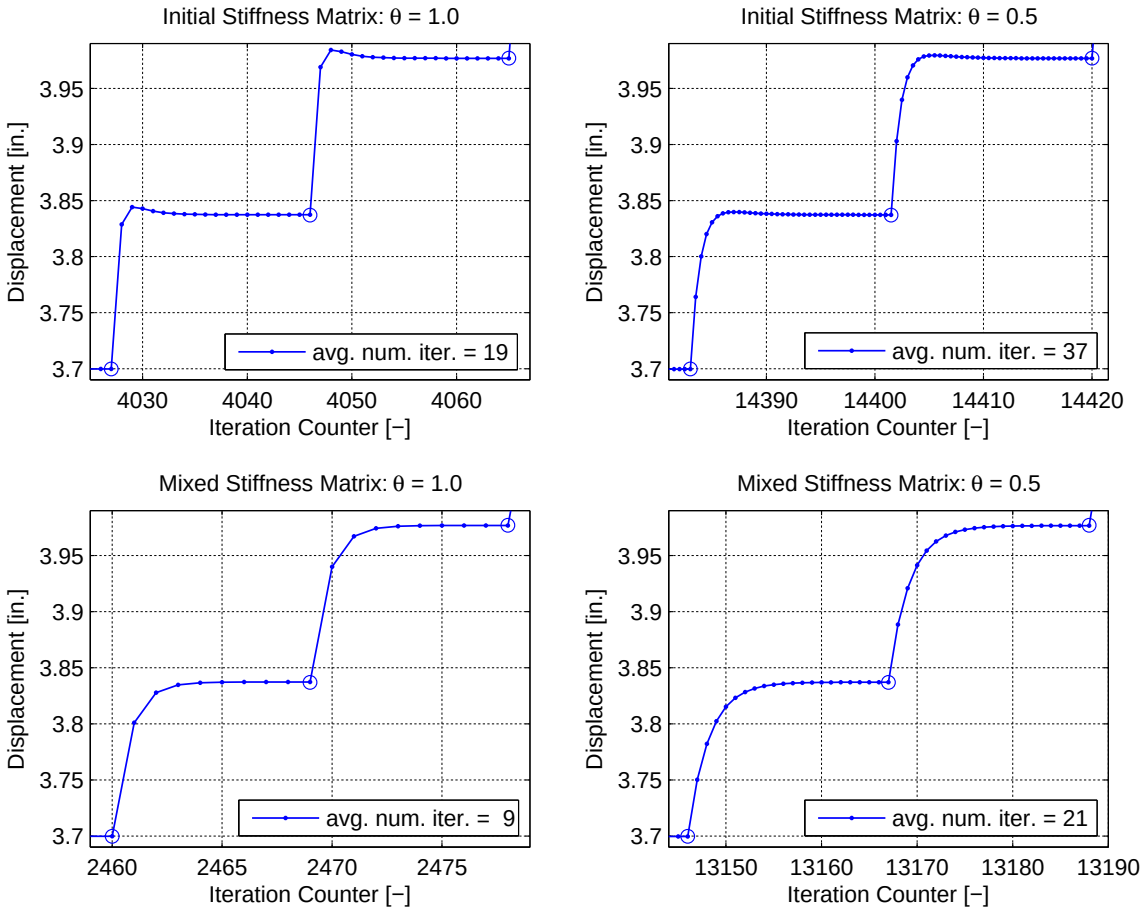


Fig. 4.8 Newton-Raphson convergence comparison for portal-frame model.

However, by utilizing the mixed Jacobian instead of the initial Jacobian in the iteration process, this problem completely disappears. The increment reduction factor in combination with the mixed stiffness matrix iterations can then further improve the smoothness of the convergence

path. A similar behavior is observed if the Krylov-Newton instead of the Newton-Raphson equilibrium solution algorithm is employed. For this specific model, the Krylov-Newton algorithm achieves significant convergence acceleration without being much affected by the increment reduction factor.

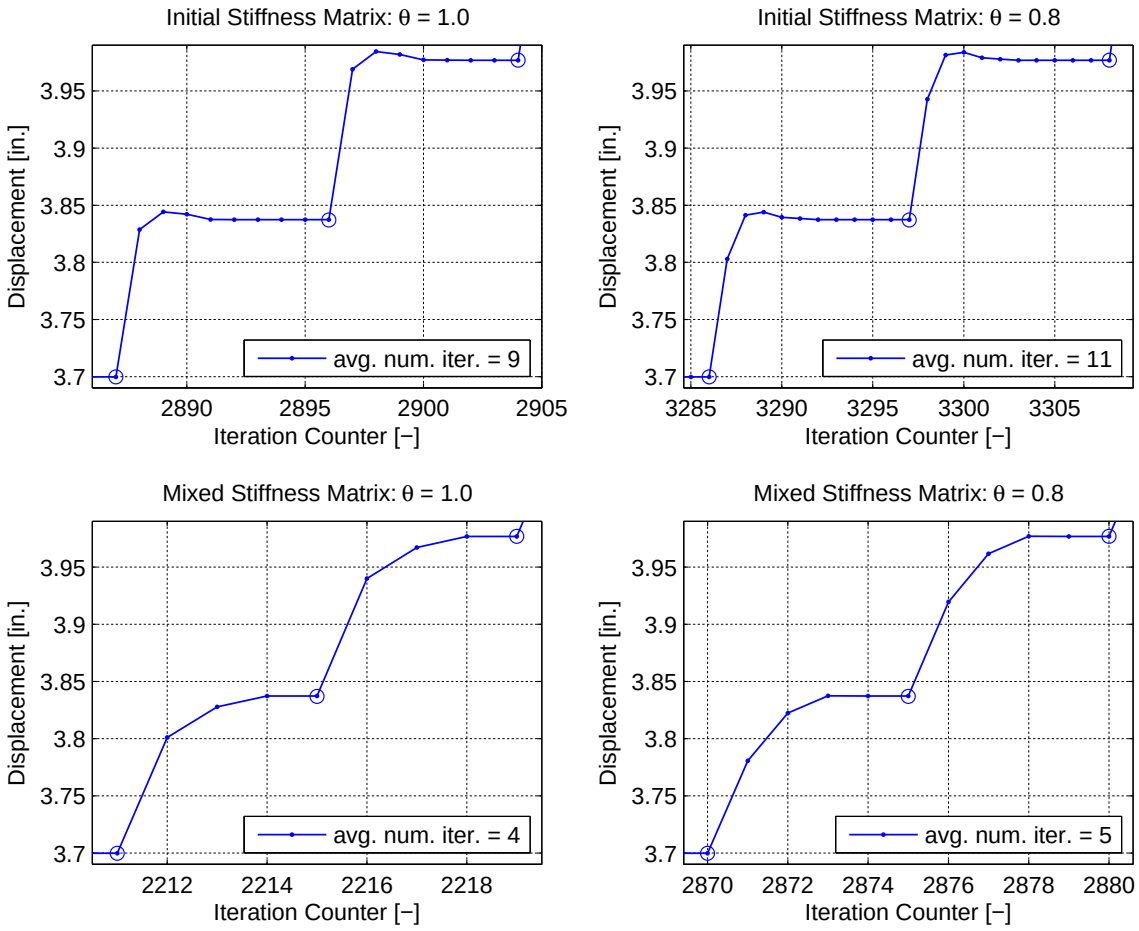


Fig. 4.9 Krylov-Newton convergence comparison for portal-frame model.

The best convergence path for the portal-frame model with its strong geometric and material nonlinearities is achieved by employing the Krylov-Newton equilibrium solution algorithm with the mixed stiffness matrix and without reduction of the increments.

In light of the special hybrid simulation requirements, the following conclusions can be drawn about the increment reduction factor implicit Newmark direct integration method. For the algorithmic parameters $\gamma = 0.5$ and $\beta = 0.25$, the average acceleration method is second-order accurate $p = 2$ and unconditionally stable independent of the amount of physical damping present. This means that requirements 1 and 3 are both satisfied. As can be seen from the

pseudo-code in Fig. 4.5, the resisting forces are required k-times according to the number of iterations necessary to reach equilibrium by satisfying the selected convergence criterion. This implies that displacement commands need to be communicated to the control system, and forces need to be measured by the data-acquisition system also k-times per integration time step. Therefore, the fewer iterations the equilibrium solution algorithm needs to execute, the better because requirement 2 asks for as few function calls as possible.

Requirement 4 is satisfied, since for both described cases the global initial stiffness matrix and the global mixed stiffness matrix require only constant, initial, experimental element stiffness matrices for assembly. Because of the initial assumption that $\gamma = 0.5$, this integration method is not capable of introducing any numerical damping to suppress higher-mode participation as suggested by requirement 5. On the other hand, for $\gamma > 0.5$ the algorithm introduces too much numerical damping, affecting the participation of the lower modes to the solution. In addition, the scheme would no longer be second-order accurate. By utilizing the described stiffness matrix and increment reduction factor modifications, it is possible to guarantee that the displacement increments calculated by the equilibrium solution algorithm are strictly increasing or strictly decreasing within an integration time step. Requirement 6 is thus satisfied.

Finally, requirement 7, which states that iterative integration methods should produce displacement increments that are as uniform as possible, is not satisfied by the modified implicit Newmark scheme. This can be seen well from the convergence plots in Figures 4.6–4.9. If the control system is allotted the same time interval to impose such inconsistent displacement increments, velocity and thus force oscillations are generated in the transfer system. To lessen these oscillations, the event-driven algorithms described in the next chapter need to adjust the time interval over which the displacement increments are imposed during simulation. Thus, the integration method, as described, only works well in combination with specialized event-driven strategies. In summary, except for the lack of algorithmic damping, the modified implicit Newmark method is an attractive integration scheme for hybrid simulation as long as it is combined with the appropriate event-driven strategies. It is recommended that this scheme be used for stiff or infinitely stiff problems that have strong material and geometric nonlinearities, and therefore, need well-converged equilibrium states at the end of the iteration process.

Constant Number of Iterations Modification (INM2):

Because the previously described modifications of the implicit Newmark method still produce nonuniform displacement increments and because the integration method in this form cannot be employed for real-time hybrid simulations, an alternative implementation of the algorithm is discussed next. The goal is to fix the number of equilibrium iterations to a constant user-specified value and to modify the displacement increments during iteration in a way that makes them more uniform. The approach of fixing the number of sub-steps or iterations was initially suggested by Dorka and Heiland (1991) and Shing et al. (1991). Shing's approach was subsequently refined by Zhong (2005). The direct integration scheme presented here is a slightly modified version of the one used in Zhong's work.

As with the first modified implicit Newmark scheme, the formulation of the Jacobian and the application of the equilibrium solution algorithms remain unchanged. In the first case, a modified Newton-Raphson algorithm with a constant initial stiffness matrix, and thus, a constant Jacobian, is utilized. In the second case, the global stiffness matrix is a nonconstant, mixed matrix, assembled from tangent and initial stiffness element matrices, and thus, the Jacobian is nonconstant as well. In order to obtain a constant number of iterations that can be user specified, a new convergence test is introduced. Instead of terminating the equilibrium solution process when the norm of a certain quantity falls below a user-specified tolerance, the equilibrium solution process is stopped after a given, constant number of iterations. The absolute energy norm (see Table 4.1) is still determined but only to provide a measure for the convergence error size. In addition, only parts of the calculated response increments are utilized to update the elements. The reduced command displacements are determined by means of Lagrange interpolation using the trial displacements of the current iteration step and the last n committed displacements. The location of the Lagrange interpolation depends on the ratio between the current iteration and the fixed number of total iterations. The polynomials used to generate the scaled displacement increments are Lagrange polynomials of first, second, or third order.

$$\Delta \mathbf{U}_{scaled}^{(k)}(x) = -\mathbf{U}_{i+1}^{(k-1)} + \sum_{j=i+1-n}^{i+1} \mathbf{U}_j \, L_{n,j}(x) \tag{4.50}$$

In the above interpolation formula $\mathbf{U}_{i+1}^{(k-1)}$ are the trial displacements at the previous iteration, $\mathbf{U}_j$ are the committed displacements at the previous time steps and the trial displacement of the current iteration, $L_{n,j}$ are the Lagrange functions of order n, and

$x = k / k_{\max} \in [0,1]$ is the interpolation location. The Lagrange functions necessary in Equation (4.50) are summarized in Table 4.2.

Table 4.2 Lagrange functions up to order 3.

$L_{1,i+1} = x$	$L_{2,i+1} = \frac{1}{2} x(x+1)$	$L_{3,i+1} = \frac{1}{6} x(x+1)(x+2)$
$L_{1,i} = -(x-1)$	$L_{2,i} = -(x-1)(x+1)$	$L_{3,i} = -\frac{1}{2}(x-1)(x+1)(x+2)$
-	$L_{2,i-1} = \frac{1}{2}(x-1)x$	$L_{3,i-1} = \frac{1}{2}(x-1)x(x+2)$
-	-	$L_{3,i-2} = -\frac{1}{6}(x-1)x(x+1)$

This reduction scheme, which is similar to the one proposed by Zhong (2005), produces displacement increments that are nearly uniform, and thus, satisfies requirement 7 for hybrid simulation.

Because the effective force vector is determined at the end of each iteration step, one additional equilibrium solution sequence is added to the integration method after the iteration process has converged. However, the elements are no longer updated with the response quantities, so that the experimental elements do not communicate these last response increments to the control system in the laboratory. The response increments are utilized only to update the response at the global degrees of freedom from which the solution process is continued in the next integration time step.

This approach achieves significantly improved convergence characteristics of the integration scheme as compared to methods without the additional equilibrium solution sequence. The second modified implicit Newmark method for hybrid simulation is summarized as pseudo-code in Fig. 4.10. As with the first modified form of the implicit Newmark method, the integration scheme is unconditionally stable if the algorithmic parameters $\gamma = 0.5$ (for second-order accuracy) and $\beta \geq 0.25$ are selected.

Initialize :

$\mathbf{U}_0$, $\dot{\mathbf{U}}_0$, $\ddot{\mathbf{U}}_0$

$\mathbf{K}_{\mathbf{eff}} = c_3 \mathbf{M} + c_2 \mathbf{C} + c_1 \mathbf{K}_{\mathbf{i}}$ *(case 1 : initial stiffness)*

Analyze :

for $(i = 0,\ i < N,\ i{+}{+})$

$$\mathbf{U}_{i+1}^{(k=1)} = \mathbf{U}_i$$

$$\dot{\mathbf{U}}_{i+1}^{(k=1)} = -\left(\frac{\gamma}{\beta} - 1\right)\dot{\mathbf{U}}_i - \Delta t\left(\frac{\gamma}{2\beta} - 1\right)\ddot{\mathbf{U}}_i$$

$$\ddot{\mathbf{U}}_{i+1}^{(k=1)} = -\frac{1}{\Delta t\,\beta}\dot{\mathbf{U}}_i - \left(\frac{1}{2\beta} - 1\right)\ddot{\mathbf{U}}_i$$

$$\mathbf{P}_{\mathbf{eff}}^{(k=1)} = \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} - \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}^{(k=1)}\right) - \mathbf{M}\,\ddot{\mathbf{U}}_{i+1}^{(k=1)} - \mathbf{C}\,\dot{\mathbf{U}}_{i+1}^{(k=1)}$$

do

$$\mathbf{K}_{\mathbf{eff}}^{(k)} = c_3 \mathbf{M} + c_2 \mathbf{C} + c_1 \mathbf{K}_{\mathbf{m}}\left(\mathbf{U}_{i+1}^{(k)}\right) \quad \textit{(case 2 : mixed stiffness)}$$

$$\Delta\mathbf{U}^{(k)} = solve\left(\mathbf{K}_{\mathbf{eff}}^{(k)}\,\Delta\mathbf{U}^{(k)} = \mathbf{P}_{\mathbf{eff}}^{(k)}\right)$$

$$\Delta\mathbf{U}_{scaled}^{(k)} = -\mathbf{U}_{i+1}^{(k-1)} + \sum_{j=i+1-n}^{i+1} \mathbf{U}_j\, L_{n,j}\left(\frac{k}{k_{\max}}\right)$$

$$\mathbf{U}_{i+1}^{(k+1)} = \mathbf{U}_{i+1}^{(k)} + c_1\,\Delta\mathbf{U}_{scaled}^{(k)}$$

$$\dot{\mathbf{U}}_{i+1}^{(k+1)} = \dot{\mathbf{U}}_{i+1}^{(k)} + c_2\,\Delta\mathbf{U}_{scaled}^{(k)}$$

$$\ddot{\mathbf{U}}_{i+1}^{(k+1)} = \ddot{\mathbf{U}}_{i+1}^{(k)} + c_3\,\Delta\mathbf{U}_{scaled}^{(k)}$$

$$\mathbf{P}_{\mathbf{eff}}^{(k+1)} = \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} - \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}^{(k+1)}\right) - \mathbf{M}\,\ddot{\mathbf{U}}_{i+1}^{(k+1)} - \mathbf{C}\,\dot{\mathbf{U}}_{i+1}^{(k+1)}$$

while $(k < k_{\max};\ k{+}{+})$

$$\mathbf{K}_{\mathbf{eff}} = c_3 \mathbf{M} + c_2 \mathbf{C} + c_1 \mathbf{K}_{\mathbf{m}}\left(\mathbf{U}_{i+1}^{(k+1)}\right) \quad \textit{(case 2 : mixed stiffness)}$$

$$\Delta\mathbf{U} = solve\left(\mathbf{K}_{\mathbf{eff}}\,\Delta\mathbf{U} = \mathbf{P}_{\mathbf{eff}}^{(k+1)}\right)$$

$$\mathbf{U}_{i+1} = \mathbf{U}_{i+1}^{(k+1)} + c_1\,\Delta\mathbf{U}$$

$$\dot{\mathbf{U}}_{i+1} = \dot{\mathbf{U}}_{i+1}^{(k)} + c_2\,\Delta\mathbf{U}$$

$$\ddot{\mathbf{U}}_{i+1} = \ddot{\mathbf{U}}_{i+1}^{(k)} + c_3\,\Delta\mathbf{U}$$

end

Fig. 4.10 Second modified D-form of implicit Newmark (INM2) method.

The progression of convergence during the equilibrium solution process is again investigated by means of the two previously described hybrid simulation models, which are analyzed by the constant-number-of-iterations implicit Newmark method.

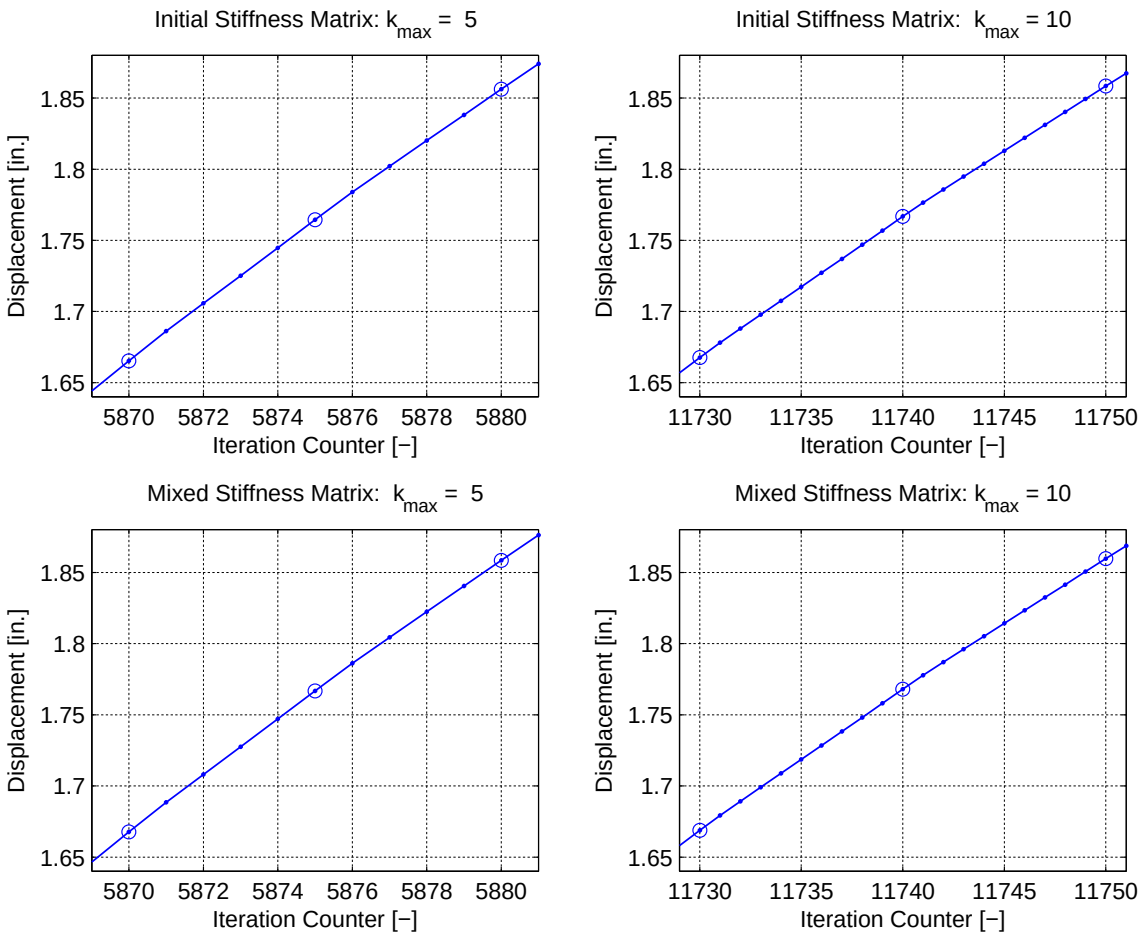


Fig. 4.11 Convergence comparison for four-story-frame model.

As can be seen from the results of the four-story-frame model (Fig. 4.11), the constant number of iterations modification achieves very smooth and uniform convergence paths. This is true for all four cases, including the initial stiffness matrix iteration, the mixed stiffness matrix iteration, a constant number of 5 increments, and a constant number of 10 increments. However, it can also be seen from the roof displacement time histories of the four-story-frame model (Fig. 4.12), that the mixed stiffness matrix iterations provide a more accurate structural response as compared to the initial stiffness matrix iterations.

In addition, the accuracy improves with larger constant numbers of iterations. The structural response, obtained by the unmodified implicit Newmark method with a very stringent displacement increment convergence test, was used as the assumed exact solution. For the most accurate solution (mixed stiffness, $k_{max} = 10$), the error in the maximal displacement is 0.05%, and the error in the residual displacement is 0.2%.

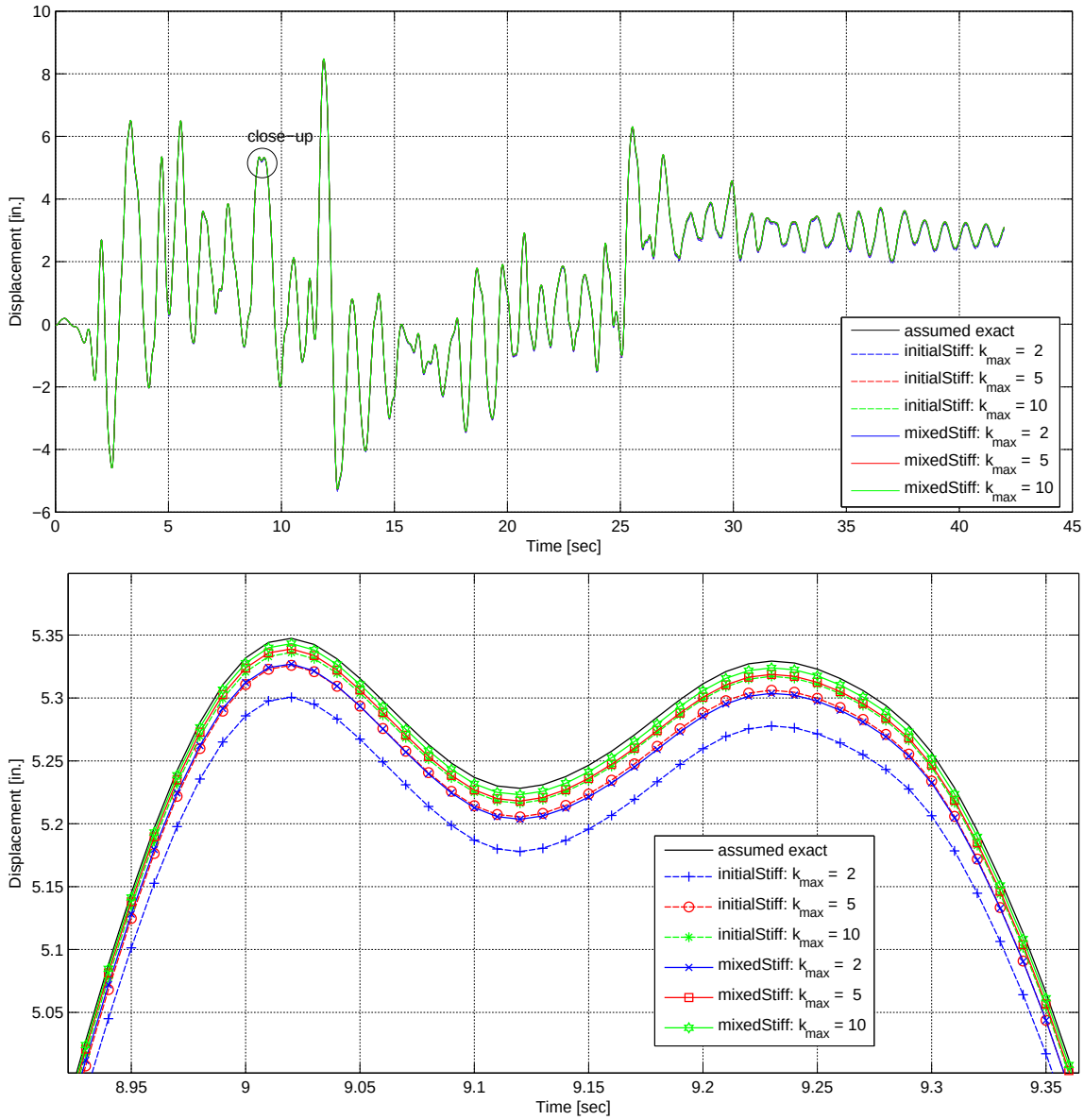


Fig. 4.12 Displacement comparison for four-story-frame model.

Similar results are obtained for the portal-frame model with its strong geometric and material nonlinearities. The convergence paths (Fig. 4.13) are all very smooth with just the slightest reduction in consistency for the cases employing the mixed Jacobian iterations. Due to the strong geometric and material nonlinearities in the portal-frame model, the trends that were observed for the four-story-frame model are even more pronounced here.

It is obvious from Fig. 4.14 that the mixed Jacobian iterations achieve superior accuracy and that such accuracy improves for larger constant numbers of iterations. For the most accurate

solution (mixed stiffness, k_{max} = 10), the error in the maximal displacement is 0.4% and the error in the residual displacement is 0.6%.

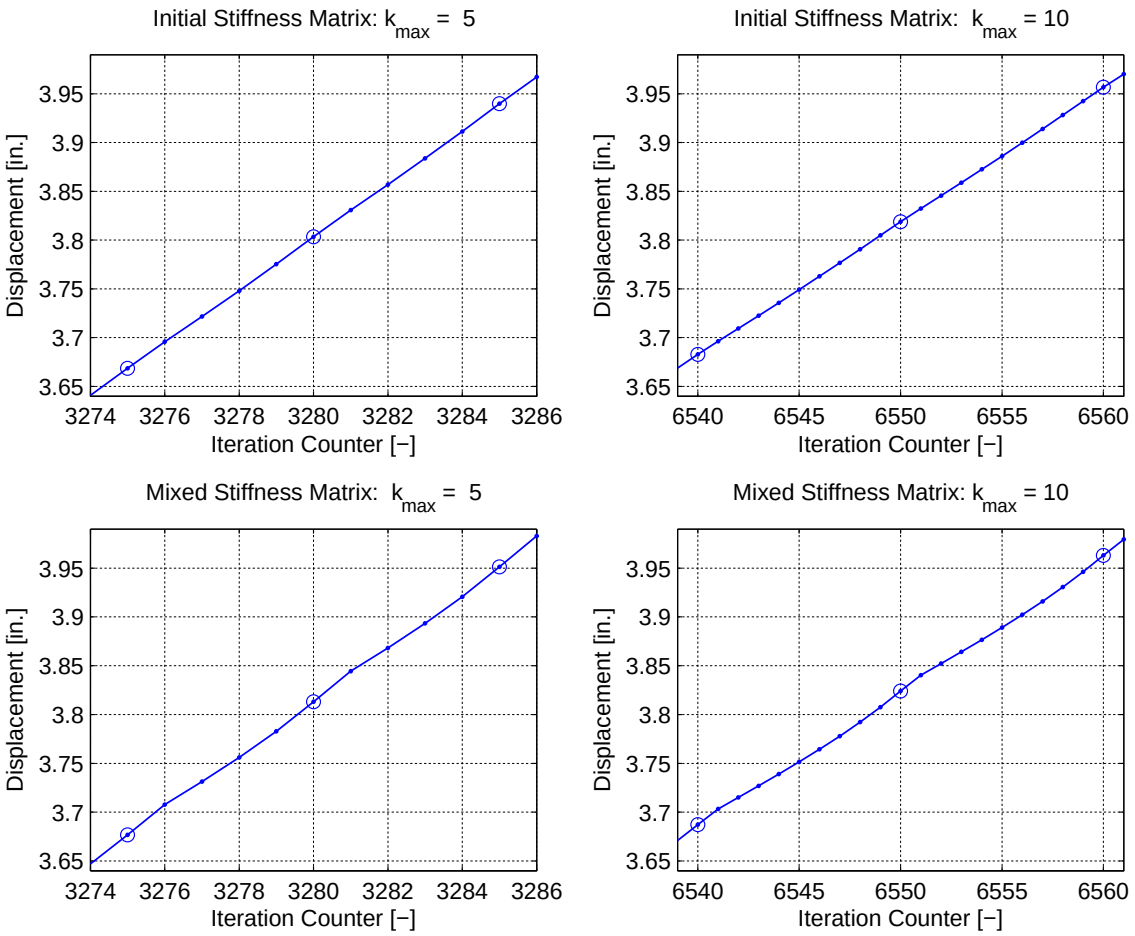


Fig. 4.13 Convergence comparison for portal-frame model.

Finally, the constant-number-of-increments implicit Newmark method is again evaluated in terms of the special hybrid simulation requirements. For the algorithmic parameters $\gamma = 0.5$ and $\beta = 0.25$, the average acceleration method is second-order accurate $p = 2$ and unconditionally stable independent of the amount of physical damping present. This means that requirements 1 and 3 are both satisfied. As can be seen from the pseudo-code in Fig. 4.10, the resisting forces are required k_{max}-times according to the constant number of iterations specified by the user. This implies that displacement commands need to be communicated to the control system and forces need to be measured by the data-acquisition system also k_{max}-times per integration time step. Requirement 4 is satisfied, since for both described cases the global initial stiffness matrix and the global mixed stiffness matrix require only constant, initial, experimental

element stiffness matrices for assembly. However, as has been shown earlier a mixed Jacobian should preferably be used. Such matrix achieves much better convergence during iteration, which improves the accuracy of the structural response significantly.

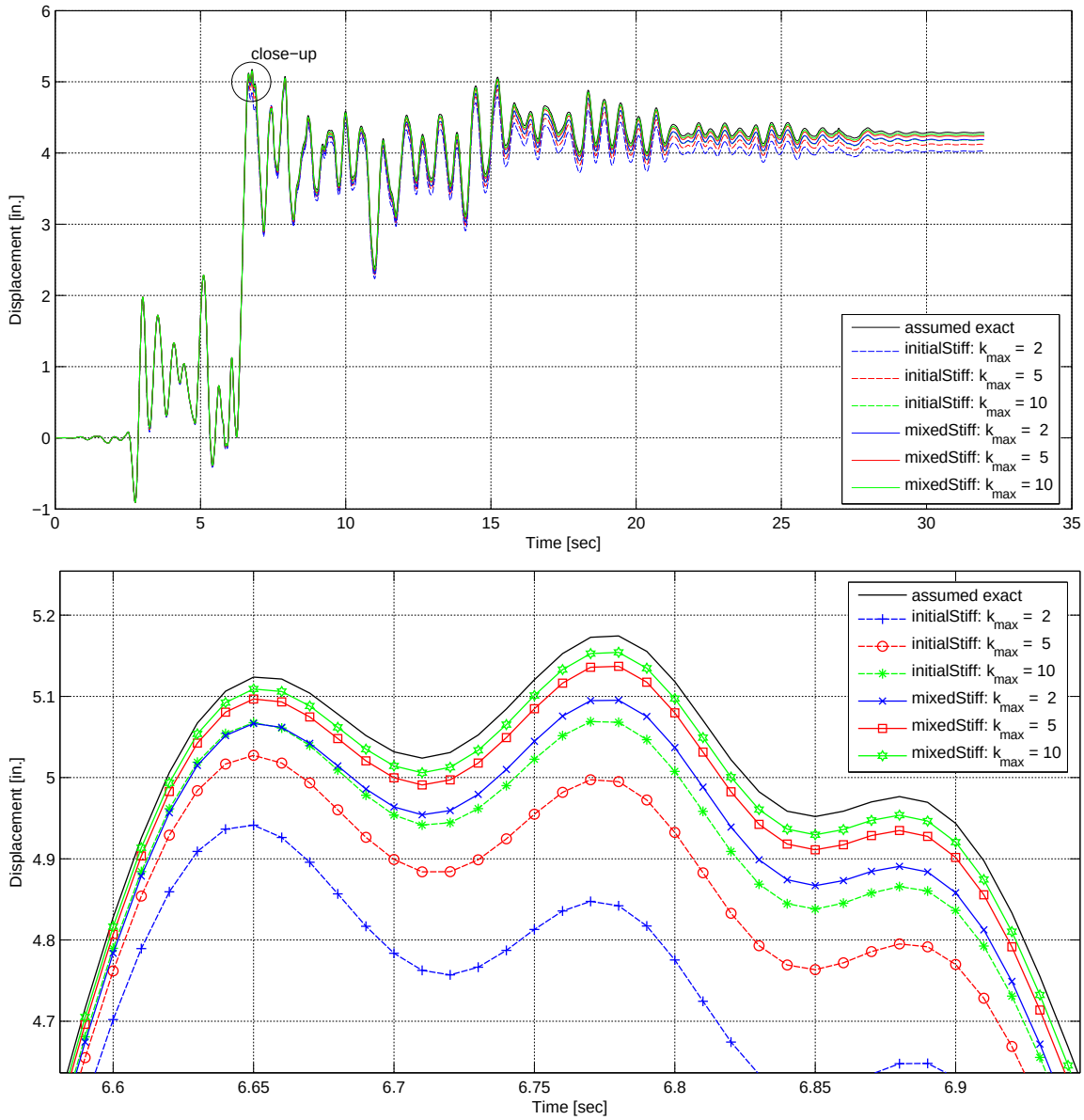


Fig. 4.14 Displacement comparison for portal-frame model.

Because of the initial assumption, that $\gamma = 0.5$, this integration method is not capable of introducing any numerical damping to suppress higher-mode participation as suggested by requirement 5. However, it is possible to obtain the desired numerical dissipation by applying the same modifications to the implicit generalized-alpha integration method, which is discussed next. By utilizing a constant number of scaled displacement increments, such increments are

strictly increasing or strictly decreasing within an integration time step. Requirement 6 is thus satisfied. Finally, requirement 7, which states that iterative integration methods should produce displacement increments that are as uniform as possible, is also satisfied due to the described Lagrange interpolation of the trial displacements. Thus, the integration method, as described, provides excellent results as long as the experimental parts of the structure do not exhibit extreme combined material and geometric nonlinearities. If the structure includes such experimental portions, it is recommended to consider the previously mentioned algorithms for estimating element tangent stiffness matrices from the experimental measurements.

4.4.4 Implicit Generalized-Alpha Methods (IGa1, IGa2)

To introduce the desired algorithmic energy dissipation, which is missing in the implicit Newmark methods, the generalized-alpha scheme developed by Chung and Hulbert (1993) is utilized again. Similarly to the explicit form of the integration method, the implicit form is second-order accurate, provides optimized dissipation characteristics of the high-frequency modes, and can have the amount of algorithmic damping specified through the spectral radius ρ_∞ at the high-frequency limit $\Omega_\infty = \omega_\infty \Delta t \to \infty$. The algorithm is optimal in the sense that it provides minimized low-frequency dissipation for a given value of high-frequency dissipation $\rho_\infty \in [0,1]$. As with the explicit form, the weighed equations of motion (4.51) are formulated in between time steps t_i and t_{i+1}, depending on the choice of the two weighing parameters α_m and α_f. In addition, Newmark's finite difference formulas (4.13) are utilized again but in their original, unchanged form.

$$\begin{aligned}
&\mathbf{M}\,\ddot{\mathbf{U}}_{i+\alpha_m} + \mathbf{C}\,\dot{\mathbf{U}}_{i+\alpha_f} + \mathbf{P}_{\mathbf{r},i+\alpha_f} = \mathbf{P}_{i+\alpha_f} - \mathbf{P}_{\mathbf{0},i+\alpha_f} \\
&\mathbf{U}_{i+1} = \mathbf{U}_i + \Delta t\,\dot{\mathbf{U}}_i + \frac{\Delta t^2}{2}\left((1-2\beta)\,\ddot{\mathbf{U}}_i + 2\beta\,\ddot{\mathbf{U}}_{i+1}\right) \\
&\dot{\mathbf{U}}_{i+1} = \dot{\mathbf{U}}_i + \Delta t\left((1-\gamma)\,\ddot{\mathbf{U}}_i + \gamma\,\ddot{\mathbf{U}}_{i+1}\right)
\end{aligned} \tag{4.51}$$

In the above system of balance equations, the accelerations and velocities in between time steps are expressed by the following weighed equations.

$$\begin{aligned}
\ddot{\mathbf{U}}_{i+\alpha_m} &= (1-\alpha_m)\,\ddot{\mathbf{U}}_i + \alpha_m\,\ddot{\mathbf{U}}_{i+1} \\
\dot{\mathbf{U}}_{i+\alpha_f} &= (1-\alpha_f)\,\dot{\mathbf{U}}_i + \alpha_f\,\dot{\mathbf{U}}_{i+1}
\end{aligned} \tag{4.52}$$

However, the internal resisting forces and the externally applied forces are expressed through any generalized quadrature rule (Erlicher et al. 2002), as explained in the section discussing the corresponding explicit scheme. For example, for the generalized trapezoidal rule, the equations take the following form.

$$\begin{aligned}\mathbf{P}_{\mathbf{r},i+\alpha_f} &= \left(1-\alpha_f\right)\mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_i\right)+\alpha_f\,\mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1}\right)\\ \mathbf{P}_{i+\alpha_f}-\mathbf{P}_{\mathbf{0},i+\alpha_f} &= \left(1-\alpha_f\right)\left(\mathbf{P}_i-\mathbf{P}_{\mathbf{0},i}\right)+\alpha_f\left(\mathbf{P}_{i+1}-\mathbf{P}_{\mathbf{0},i+1}\right)\end{aligned} \tag{4.53}$$

If the generalized midpoint rule is employed instead, the following equations are utilized.

$$\begin{aligned}\mathbf{P}_{\mathbf{r},i+\alpha_f} &= \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+\alpha_f}\right)=\mathbf{P}_{\mathbf{r}}\left(\left(1-\alpha_f\right)\mathbf{U}_i+\alpha_f\mathbf{U}_{i+1}\right)\\ \mathbf{P}_{i+\alpha_f}-\mathbf{P}_{\mathbf{0},i+\alpha_f} &= \mathbf{P}\left(t_{i+\alpha_f}\right)-\mathbf{P}_{\mathbf{0}}\left(t_{i+\alpha_f}\right)\end{aligned} \tag{4.54}$$

Because the resisting forces are treated implicitly here, the displacements at the new time step are being used to evaluate the weighed forces provided by the above generalized trapezoidal and midpoint formulas.

As with the implicit Newmark method, the same two approaches can be employed to solve the three equations in (4.51) for the three unknown response quantities. Since the D-form implementation again encounters fewer convergence issues than the A-form implementation, only the D-form approach is modified herein for hybrid simulation. The two modifications that were discussed with the implicit Newmark method can be applied to the implicit generalized-alpha method in the same manner. However, only the constant number of iterations modification is derived here. For the increment reduction factor modification, please refer to the section explaining the implicit Newmark method.

Constant Number of Iterations Modification (IGα2):

In order to derive the D-form, Newmark's finite difference formulae (4.51) are again reformulated to obtain expressions for velocities and accelerations at t_{i+1}.

$$\begin{aligned}\dot{\mathbf{U}}_{i+1} &= \frac{\gamma}{\Delta t\,\beta}\left(\mathbf{U}_{i+1}-\mathbf{U}_i\right)-\left(\frac{\gamma}{\beta}-1\right)\dot{\mathbf{U}}_i-\Delta t\left(\frac{\gamma}{2\beta}-1\right)\ddot{\mathbf{U}}_i\\ \ddot{\mathbf{U}}_{i+1} &= \frac{1}{\Delta t^2\beta}\left(\mathbf{U}_{i+1}-\mathbf{U}_i\right)-\frac{1}{\Delta t\,\beta}\dot{\mathbf{U}}_i-\left(\frac{1}{2\beta}-1\right)\ddot{\mathbf{U}}_i\end{aligned} \tag{4.55}$$

Substitution of these velocities and accelerations into the weighed balance Equations (4.51) and equating such equations to zero yields a nonlinear system of equations, in which the unknowns are the displacements at the new time step $t_i + \Delta t$.

$$\begin{aligned} F\left(\mathbf{U}_{i+1}\right) = &\left(1-\alpha_m\right)\mathbf{M}\,\ddot{\mathbf{U}}_i + \alpha_m \mathbf{M}\,\ddot{\mathbf{U}}_{i+1} + \left(1-\alpha_f\right)\mathbf{C}\,\dot{\mathbf{U}}_i + \alpha_f \mathbf{C}\,\dot{\mathbf{U}}_{i+1} \\ &+ \mathbf{P}_{\mathbf{r},i+\alpha_f}\left(\mathbf{U}_{i+1}\right) - \mathbf{P}_{i+\alpha_f} + \mathbf{P}_{\mathbf{0},i+\alpha_f} = 0 \end{aligned} \tag{4.56}$$

In order to solve such system of nonlinear equations for the unknown displacements, the well-known iterative Newton-Raphson algorithm is employed.

$$DF\left(\mathbf{U}_{i+1}^{(k)}\right)\Delta\mathbf{U}^{(k)} = -F\left(\mathbf{U}_{i+1}^{(k)}\right) \tag{4.57}$$

The Jacobian of the vector function DF on the left-hand side and the negative vector function F on the right-hand side both evaluated at the displacements of the current iteration are equivalent to an effective stiffness matrix and an effective force vector (also called unbalanced force vector), respectively.

$$\mathbf{K}_{\mathbf{eff}}^{(k)}\,\Delta\mathbf{U}^{(k)} = \mathbf{P}_{\mathbf{eff}}^{(k)} \tag{4.58}$$

In the above linear system of equations, the effective stiffness matrix and the effective force vector, which depend on the iteration k, take the following form.

$$\begin{aligned} \mathbf{K}_{\mathbf{eff}}^{(k)} &= \alpha_m c_3\,\mathbf{M} + \alpha_f c_2\,\mathbf{C} + \alpha_f c_1\,\mathbf{K}_{\mathbf{t}}\left(\mathbf{U}_{i+1}^{(k)}\right) \\ \mathbf{P}_{\mathbf{eff}}^{(k)} &= \mathbf{P}_{i+\alpha_f} - \mathbf{P}_{\mathbf{0},i+\alpha_f} - \mathbf{P}_{\mathbf{r},i+\alpha_f}\left(\mathbf{U}_{i+1}^{(k)}\right) \\ &\quad -\mathbf{M}\left[\left(1-\alpha_m\right)\ddot{\mathbf{U}}_i + \alpha_m\ddot{\mathbf{U}}_{i+1}\right] - \mathbf{C}\left[\left(1-\alpha_f\right)\dot{\mathbf{U}}_i + \alpha_f\dot{\mathbf{U}}_{i+1}\right] \end{aligned} \tag{4.59}$$

where $c_1 = 1$, $c_2 = \gamma/(\Delta t\,\beta)$, and $c_3 = 1/(\Delta t^2\beta)$. Since the Newton-Raphson algorithm guarantees convergence only if the initial approximation of the displacement vector is sufficiently close to the actual solution, the converged displacement vector at time t_i is used as an initial approximation for the sought displacement vector at time $t_i + \Delta t$.

$$\begin{aligned} \mathbf{U}_{i+1}^{(k=1)} &= \mathbf{U}_i \\ \dot{\mathbf{U}}_{i+1}^{(k=1)} &= -\left(\frac{\gamma}{\beta}-1\right)\dot{\mathbf{U}}_i - \Delta t\left(\frac{\gamma}{2\beta}-1\right)\ddot{\mathbf{U}}_i \\ \ddot{\mathbf{U}}_{i+1}^{(k=1)} &= -\frac{1}{\Delta t\,\beta}\dot{\mathbf{U}}_i - \left(\frac{1}{2\beta}-1\right)\ddot{\mathbf{U}}_i \end{aligned} \tag{4.60}$$

Once Equation (4.58) has been solved for the displacement increments at iteration k, such increments are scaled using Lagrange polynomials of first, second, or third order. As before, the location of the Lagrange interpolation depends on the ratio between the current iteration and the fixed number of total iterations (4.50). The scaled displacement increments are then utilized to update the response quantities such as displacements, velocities, and accelerations as follows.

$$\begin{aligned}
\mathbf{U}_{i+1}^{(k+1)} &= \mathbf{U}_{i+1}^{(k)} + c_1\,\Delta\mathbf{U}_{scaled}^{(k)} \\
\dot{\mathbf{U}}_{i+1}^{(k+1)} &= \dot{\mathbf{U}}_{i+1}^{(k)} + c_2\,\Delta\mathbf{U}_{scaled}^{(k)} \\
\ddot{\mathbf{U}}_{i+1}^{(k+1)} &= \ddot{\mathbf{U}}_{i+1}^{(k)} + c_3\,\Delta\mathbf{U}_{scaled}^{(k)}
\end{aligned} \tag{4.61}$$

The iterations are repeated until the user-specified, fixed number of total iterations is reached. As with the implicit Newmark scheme, the effective force vector is determined at the end of each iteration step, and thus, one additional equilibrium solution sequence is added to the integration method after the iteration process has ended. However, the elements are no longer updated with the response quantities, so that the experimental elements do not communicate these last response increments to the control system in the laboratory.

The response increments are utilized to update only the response at the global degrees of freedom from which the solution process is continued in the next integration time step. The repetition of this procedure for the total number of N time steps, with $k_{\max}$ iterations in each such step, yields the sought solution. The modified D-form of the implicit generalized-alpha method is summarized as pseudo-code in Fig. 4.15.

To maximize the high-frequency algorithmic energy dissipation while minimizing low-frequency dissipation and maintaining second-order accuracy, Chung and Hulbert (1993) derived the following relationships among the algorithmic parameters β and γ, the weighing parameters, α_m and α_f, and the high-frequency spectral radius, $0 \le \rho_\infty \le 1$.

$$\begin{aligned}
\alpha_m &= \frac{2-\rho_\infty}{1+\rho_\infty} && \in [2.0,\ 0.5] \\
\alpha_f &= \frac{1}{1+\rho_\infty} && \in [1.0,\ 0.5] \\
\beta &= \frac{1}{(1+\rho_\infty)^2} && \in [1,\ 0.25] \\
\gamma &= 0.5+\alpha_m-\alpha_f && \in [1.5,\ 0.5]
\end{aligned} \tag{4.62}$$

Initialize :

$\mathbf{U}_0$, $\dot{\mathbf{U}}_0$, $\ddot{\mathbf{U}}_0$

$\mathbf{K}_{\mathbf{eff}} = \alpha_m c_3 \mathbf{M} + \alpha_f c_2 \mathbf{C} + \alpha_f c_1 \mathbf{K}_{\mathbf{i}}$ *(case 1 : initial stiffness)*

Analyze :

for $(i = 0,\ i < N,\ i{+}{+})$

$\mathbf{U}_{i+1}^{(k=1)} = \mathbf{U}_i$

$\dot{\mathbf{U}}_{i+1}^{(k=1)} = -\left(\frac{\gamma}{\beta} - 1\right)\dot{\mathbf{U}}_i - \Delta t\left(\frac{\gamma}{2\beta} - 1\right)\ddot{\mathbf{U}}_i$

$\ddot{\mathbf{U}}_{i+1}^{(k=1)} = -\frac{1}{\Delta t\,\beta}\dot{\mathbf{U}}_i - \left(\frac{1}{2\beta} - 1\right)\ddot{\mathbf{U}}_i$

$\mathbf{P}_{\mathbf{eff}}^{(k=1)} = \mathbf{P}_{i+\alpha_f} - \mathbf{P}_{\mathbf{0},i+\alpha_f} - \mathbf{P}_{\mathbf{r},i+\alpha_f}\left(\mathbf{U}_{i+1}^{(k=1)}\right)$ *(Eq. 4.53 or Eq. 4.54)*

$-\mathbf{M}\left[(1-\alpha_m)\ddot{\mathbf{U}}_i + \alpha_m \ddot{\mathbf{U}}_{i+1}^{(k=1)}\right] - \mathbf{C}\left[(1-\alpha_f)\dot{\mathbf{U}}_i + \alpha_f \dot{\mathbf{U}}_{i+1}^{(k=1)}\right]$

do

$\mathbf{K}_{\mathbf{eff}}^{(k)} = \alpha_m c_3 \mathbf{M} + \alpha_f c_2 \mathbf{C} + \alpha_f c_1 \mathbf{K}_{\mathbf{m}}\left(\mathbf{U}_{i+1}^{(k)}\right)$ *(case 2 : mixed stiffness)*

$\Delta\mathbf{U}^{(k)} = solve\left(\mathbf{K}_{\mathbf{eff}}^{(k)}\,\Delta\mathbf{U}^{(k)} = \mathbf{P}_{\mathbf{eff}}^{(k)}\right)$

$\Delta\mathbf{U}_{scaled}^{(k)} = -\mathbf{U}_{i+1}^{(k-1)} + \sum_{j=i+1-n}^{i+1} \mathbf{U}_j\, L_{n,j}\left(\frac{k}{k_{\max}}\right)$

$\mathbf{U}_{i+1}^{(k+1)} = \mathbf{U}_{i+1}^{(k)} + c_1\,\Delta\mathbf{U}_{scaled}^{(k)}$

$\dot{\mathbf{U}}_{i+1}^{(k+1)} = \dot{\mathbf{U}}_{i+1}^{(k)} + c_2\,\Delta\mathbf{U}_{scaled}^{(k)}$

$\ddot{\mathbf{U}}_{i+1}^{(k+1)} = \ddot{\mathbf{U}}_{i+1}^{(k)} + c_3\,\Delta\mathbf{U}_{scaled}^{(k)}$

$\mathbf{P}_{\mathbf{eff}}^{(k+1)} = \mathbf{P}_{i+\alpha_f} - \mathbf{P}_{\mathbf{0},i+\alpha_f} - \mathbf{P}_{\mathbf{r},i+\alpha_f}\left(\mathbf{U}_{i+1}^{(k+1)}\right)$ *(Eq. 4.53 or Eq. 4.54)*

$-\mathbf{M}\left[(1-\alpha_m)\ddot{\mathbf{U}}_i + \alpha_m \ddot{\mathbf{U}}_{i+1}^{(k+1)}\right] - \mathbf{C}\left[(1-\alpha_f)\dot{\mathbf{U}}_i + \alpha_f \dot{\mathbf{U}}_{i+1}^{(k+1)}\right]$

while $(k < k_{\max};\ k{+}{+})$

$\mathbf{K}_{\mathbf{eff}} = \alpha_m c_3 \mathbf{M} + \alpha_f c_2 \mathbf{C} + \alpha_f c_1 \mathbf{K}_{\mathbf{m}}\left(\mathbf{U}_{i+1}^{(k+1)}\right)$ *(case 2 : mixed stiffness)*

$\Delta\mathbf{U} = solve\left(\mathbf{K}_{\mathbf{eff}}\,\Delta\mathbf{U} = \mathbf{P}_{\mathbf{eff}}^{(k+1)}\right)$

$\mathbf{U}_{i+1} = \mathbf{U}_{i+1}^{(k+1)} + c_1\,\Delta\mathbf{U}$

$\dot{\mathbf{U}}_{i+1} = \dot{\mathbf{U}}_{i+1}^{(k)} + c_2\,\Delta\mathbf{U}$

$\ddot{\mathbf{U}}_{i+1} = \ddot{\mathbf{U}}_{i+1}^{(k)} + c_3\,\Delta\mathbf{U}$

end

Fig. 4.15 Second modified D-form of implicit generalized-alpha (IGα2) method.

The resulting implicit generalized-alpha direct integration method is a one-parameter (ρ_∞) scheme, which is unconditionally stable and second-order $p=2$ accurate for displacements and velocities and first-order accurate for accelerations. Furthermore, the method satisfies the same special hybrid simulation requirements as the constant number of iterations implicit Newmark scheme, with the important advantage that an optimized, user-adjustable, high-frequency, algorithmic energy dissipation is provided. Thus, the constant number of iterations implicit generalized-alpha method additionally satisfies requirement 5 and should therefore be the preferred scheme for hybrid simulation as compared to the corresponding implicit Newmark method.

4.4.5 Generalized-Alpha-OS Method (GaOS1)

The generalized-alpha-OS direct integration method is comparable to the alpha-OS scheme originally developed by Nakashima (1988). The algorithm is a combination of the operator-splitting technique proposed by Hughes et al. (1979) and the generalized-alpha method developed by Chung and Hulbert (1993). Hence, this alternative approach introduced here is different than the one proposed by Bonelli and Bursi (2004). The resulting integration scheme is a predictor-one-corrector method that does not require iterative equilibrium solution algorithms and that inherits the optimized algorithmic dissipation characteristics of the family of generalized-alpha methods. The operator-splitting technique is utilized to approximate the deviation of the structural response from linearity, meaning that the nonlinear resisting forces are expressed as the difference between the corresponding linear-elastic resisting forces and an approximate corrective part. The advantage of the class of alpha-OS integration methods is that unconditional stability can be achieved without the need to employ computationally expensive, iterative equilibrium solution algorithms.

The derivation of the algorithm starts from the same equilibrium and finite-difference formulae as the explicit generalized-alpha method.

$$\begin{aligned}
&\mathbf{M}\,\ddot{\mathbf{U}}_{i+\alpha_m} + \mathbf{C}\,\dot{\mathbf{U}}_{i+\alpha_f} + \mathbf{P}_{\mathbf{r},i+\alpha_f} = \mathbf{P}_{i+\alpha_f} - \mathbf{P}_{\mathbf{0},i+\alpha_f} \\
&\mathbf{U}_{i+1} = \mathbf{U}_i + \Delta t\,\dot{\mathbf{U}}_i + \frac{\Delta t^2}{2}(1-2\beta)\,\ddot{\mathbf{U}}_i + \Delta t^2\beta\,\ddot{\mathbf{U}}_{i+1} = \tilde{\mathbf{U}}_{i+1} + \Delta t^2\beta\,\ddot{\mathbf{U}}_{i+1} \\
&\dot{\mathbf{U}}_{i+1} = \dot{\mathbf{U}}_i + \Delta t(1-\gamma)\,\ddot{\mathbf{U}}_i + \Delta t\,\gamma\,\ddot{\mathbf{U}}_{i+1} = \dot{\tilde{\mathbf{U}}}_{i+1} + \Delta t\,\gamma\,\ddot{\mathbf{U}}_{i+1}
\end{aligned} \tag{4.63}$$

Utilizing the operator-splitting technique, the nonlinear internal resisting forces on the left-hand side of the equilibrium equations in (4.63) are approximated as follows.

$$\mathbf{P}_{\mathbf{r},i+\alpha_f} \cong \mathbf{K}_\mathbf{i}\,\mathbf{U}_{i+\alpha_f} - \left(\mathbf{K}_\mathbf{i}\,\tilde{\mathbf{U}}_{i+\alpha_f} - \tilde{\mathbf{P}}_{\mathbf{r},i+\alpha_f}\right) \tag{4.64}$$

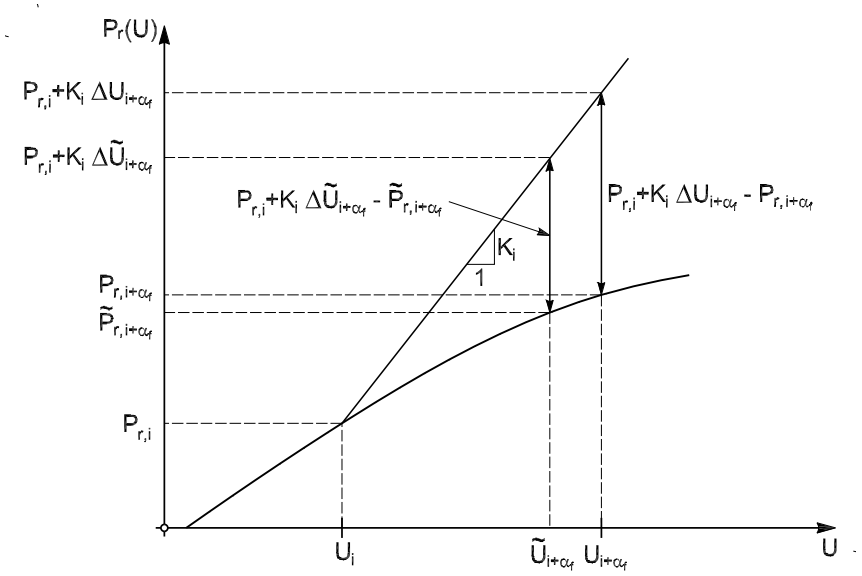


Fig. 4.16 Approximation of nonlinear resisting force using $\mathbf{K_i}$.

As can be seen from the figure above, the operator-splitting technique assumes that the difference between the elastic and the nonlinear resisting forces at the predictor displacements $\tilde{\mathbf{U}}_{i+\alpha_f}$ is approximately equal to the difference between the elastic and the nonlinear resisting forces at the new displacements $\mathbf{U}_{i+\alpha_f}$. Substitution of (4.64) into the balance Equation (4.63) leads to the following approximate equilibrium equations.

$$\mathbf{M}\,\ddot{\mathbf{U}}_{i+\alpha_m} + \mathbf{C}\,\dot{\mathbf{U}}_{i+\alpha_f} + \mathbf{K}_\mathbf{i}\left(\mathbf{U}_{i+\alpha_f} - \tilde{\mathbf{U}}_{i+\alpha_f}\right) + \tilde{\mathbf{P}}_{\mathbf{r},i+\alpha_f} = \mathbf{P}_{i+\alpha_f} - \mathbf{P}_{\mathbf{0},i+\alpha_f} \tag{4.65}$$

In the above equation, the accelerations, velocities, and displacements in between time steps are expressed by the following weighed equations.

$$\begin{aligned}
\ddot{\mathbf{U}}_{i+\alpha_m} &= (1-\alpha_m)\,\ddot{\mathbf{U}}_i + \alpha_m\,\ddot{\mathbf{U}}_{i+1} \\
\dot{\mathbf{U}}_{i+\alpha_f} &= (1-\alpha_f)\,\dot{\mathbf{U}}_i + \alpha_f\,\dot{\mathbf{U}}_{i+1} \\
\mathbf{U}_{i+\alpha_f} &= (1-\alpha_f)\,\mathbf{U}_i + \alpha_f\,\mathbf{U}_{i+1} \\
\tilde{\mathbf{U}}_{i+\alpha_f} &= (1-\alpha_f)\,\tilde{\mathbf{U}}_i + \alpha_f\,\tilde{\mathbf{U}}_{i+1}
\end{aligned} \tag{4.66}$$

Similarly to the previously discussed integration methods, the internal resisting forces at the predictor displacements and the externally applied forces are expressed in terms of any

generalized quadrature rule (Erlicher et al. 2002). For example, for the generalized trapezoidal rule, the equations take the following form.

$$\begin{aligned}\tilde{\mathbf{P}}_{\mathbf{r},i+\alpha_f} &= \left(1-\alpha_f\right)\mathbf{P}_{\mathbf{r}}\left(\tilde{\mathbf{U}}_i\right)+\alpha_f\,\mathbf{P}_{\mathbf{r}}\left(\tilde{\mathbf{U}}_{i+1}\right)\\ \mathbf{P}_{i+\alpha_f}-\mathbf{P}_{\mathbf{0},i+\alpha_f} &= \left(1-\alpha_f\right)\left(\mathbf{P}_i-\mathbf{P}_{\mathbf{0},i}\right)+\alpha_f\left(\mathbf{P}_{i+1}-\mathbf{P}_{\mathbf{0},i+1}\right)\end{aligned} \tag{4.67}$$

If the generalized midpoint rule is employed instead, the following equations are utilized.

$$\begin{aligned}\tilde{\mathbf{P}}_{\mathbf{r},i+\alpha_f} &= \mathbf{P}_{\mathbf{r}}\left(\tilde{\mathbf{U}}_{i+\alpha_f}\right)=\mathbf{P}_{\mathbf{r}}\left(\left(1-\alpha_f\right)\tilde{\mathbf{U}}_i+\alpha_f\tilde{\mathbf{U}}_{i+1}\right)\\ \mathbf{P}_{i+\alpha_f}-\mathbf{P}_{\mathbf{0},i+\alpha_f} &= \mathbf{P}\left(t_{i+\alpha_f}\right)-\mathbf{P}_{\mathbf{0}}\left(t_{i+\alpha_f}\right)\end{aligned} \tag{4.68}$$

Next, the velocities and accelerations at t_{i+1} are expressed in terms of the predictor displacements.

$$\begin{aligned}\dot{\mathbf{U}}_{i+1} &= \frac{\gamma}{\Delta t\,\beta}\left(\mathbf{U}_{i+1}-\tilde{\mathbf{U}}_{i+1}\right)+\dot{\mathbf{U}}_i+\Delta t\left(1-\gamma\right)\ddot{\mathbf{U}}_i\\ \ddot{\mathbf{U}}_{i+1} &= \frac{1}{\Delta t^2\beta}\left(\mathbf{U}_{i+1}-\tilde{\mathbf{U}}_{i+1}\right)\end{aligned} \tag{4.69}$$

Substitution of the velocities and acceleration expressions into (4.65) yields the following system of linear equations, which needs to be solved for the displacement increments $\Delta\mathbf{U}$ between the predicted and the new displacements.

$$\mathbf{K}_{\mathbf{eff}}\,\Delta\mathbf{U}=\mathbf{P}_{\mathbf{eff}} \tag{4.70}$$

In the above linear system of equations, the effective stiffness matrix and the effective force vector take the following form.

$$\begin{aligned}\mathbf{K}_{\mathbf{eff}} &= \alpha_m c_3\,\mathbf{M}+\alpha_f c_2\,\mathbf{C}+\alpha_f c_1\,\mathbf{K}_{\mathbf{i}}\\ \mathbf{P}_{\mathbf{eff}} &= \mathbf{P}_{i+\alpha_f}-\mathbf{P}_{\mathbf{0},i+\alpha_f}-\left[\tilde{\mathbf{P}}_{\mathbf{r},i+\alpha_f}+\mathbf{K}_{\mathbf{i}}\left(1-\alpha_f\right)\left(\mathbf{U}_i-\tilde{\mathbf{U}}_i\right)\right]\\ &\quad -\mathbf{M}\left(1-\alpha_m\right)\ddot{\mathbf{U}}_i-\mathbf{C}\left[\left(1-\alpha_f\right)\dot{\mathbf{U}}_i+\alpha_f\dot{\tilde{\mathbf{U}}}_{i+1}\right]\end{aligned} \tag{4.71}$$

where $c_1=1$, $c_2=\gamma/(\Delta t\,\beta)$, and $c_3=1/(\Delta t^2\beta)$. Once Equation (4.70) has been solved for the displacement increments, the response quantities such as displacements, velocities, and accelerations are updated as follows.

$$\begin{aligned}\mathbf{U}_{i+1} &= \tilde{\mathbf{U}}_{i+1}+c_1\,\Delta\mathbf{U}\\ \dot{\mathbf{U}}_{i+1} &= \dot{\tilde{\mathbf{U}}}_{i+1}+c_2\,\Delta\mathbf{U}\\ \ddot{\mathbf{U}}_{i+1} &= c_3\,\Delta\mathbf{U}\end{aligned} \tag{4.72}$$

Initialize:

$\mathbf{U}_0$, $\dot{\mathbf{U}}_0$, $\ddot{\mathbf{U}}_0$

$\mathbf{K}_{\text{eff}} = \alpha_m c_3 \mathbf{M} + \alpha_f c_2 \mathbf{C} + \alpha_f c_1 \mathbf{K}_{\mathbf{i}}$ *(factorized storage)*

Analyze:

for $(i = 0,\ i < N,\ i{+}{+})$

$$\tilde{\mathbf{U}}_{i+1} = \mathbf{U}_i + \Delta t\, \dot{\mathbf{U}}_i + \frac{\Delta t^2}{2}(1-2\beta)\, \ddot{\mathbf{U}}_i$$

$$\dot{\tilde{\mathbf{U}}}_{i+1} = \dot{\mathbf{U}}_i + \Delta t(1-\gamma)\, \ddot{\mathbf{U}}_i$$

$$\mathbf{P}_{\text{eff}} = \mathbf{P}_{i+\alpha_f} - \mathbf{P}_{\mathbf{0},i+\alpha_f} - \tilde{\mathbf{P}}_{\mathbf{r},i+\alpha_f} \qquad \textit{(Eq. 4.67 or Eq. 4.68)}$$

$$-\mathbf{K}_{\mathbf{i}}(1-\alpha_f)(\mathbf{U}_i - \tilde{\mathbf{U}}_i) - \mathbf{M}(1-\alpha_m)\ddot{\mathbf{U}}_i - \mathbf{C}\left[(1-\alpha_f)\dot{\mathbf{U}}_i + \alpha_f \dot{\tilde{\mathbf{U}}}_{i+1}\right]$$

$$\Delta\mathbf{U} = solve(\mathbf{K}_{\text{eff}}\, \Delta\mathbf{U} = \mathbf{P}_{\text{eff}})$$

$$\mathbf{U}_{i+1} = \tilde{\mathbf{U}}_{i+1} + c_1\, \Delta\mathbf{U}$$

$$\dot{\mathbf{U}}_{i+1} = \dot{\tilde{\mathbf{U}}}_{i+1} + c_2\, \Delta\mathbf{U}$$

$$\ddot{\mathbf{U}}_{i+1} = c_3\, \Delta\mathbf{U}$$

end

Fig. 4.17 Generalized-alpha-OS (GαOS1) method.

Repetition of this procedure for the total number of N time steps yields the sought solution. The generalized-alpha-OS integration method can thus be summarized as pseudo-code as shown above in Fig. 4.17.

With the selection of the integration scheme's parameter according to Equation (4.62), the method is second-order accurate $p = 2$ for displacements and velocities and first-order accurate for accelerations and therefore satisfies requirement 1. As long as at any time the predictor stiffness is larger than the tangent stiffness, the generalized-alpha-OS method is unconditionally stable (Combescure and Pegon 1997), and thus, satisfies requirement 3. This means that $\delta\mathbf{K} = \mathbf{K}_{\mathbf{i}} - \mathbf{K}_{\mathbf{t}}$ must be positive semi-definite. Thus, the generalized-alpha-OS method cannot be employed to analyze structures that exhibit hardening material behaviors or stiffening due to geometric nonlinearities. Furthermore, the method can also be rendered conditionally stable if geometric nonlinearities are accounted for by the large-displacement, moderate deformations, corotational transformation. This is further explained at the end of this section. As can be seen from the pseudo-code in Fig. 4.17, the resisting forces necessary for assembling the effective force vector are only required once per time step. Depending on the

employed generalized quadrature rule, the resisting forces are acquired either at the predictor displacements, or at weighed displacements between the last and current predictor displacements. In short, this means that no more than one force acquisition from the experimental portion of the structure is necessary per integration step. Requirement 2 is thus satisfied. Requirement 4 is satisfied, since the global initial stiffness matrix requires only constant, initial, experimental element stiffness matrices for assembly. Due to the nature of the method, requirement 5 is satisfied. Specifically, the generalized-alpha-OS integration scheme provides optimal dissipation of the high-frequency modes, while the amount of algorithmic damping can be specified through the high-frequency spectral radius ρ_∞. In addition, the accuracy of the method is only slightly lowered by the numerical energy dissipation, so that the scheme remains second-order accurate. Finally, requirements 6 and 7 do not apply here, since the generalized-alpha-OS method is not an iterative solution scheme. Because of the approximation that is used to express the resting forces in the equations of motion, the algorithm cannot provide very accurate results for highly nonlinear structures. However, as long as the analyzed structure does not exhibit severe geometric nonlinearities; the generalized-alpha-OS method is a very attractive integration scheme for hybrid simulation.

The possible loss of the unconditional stability of the generalized-alpha-OS method in combination with the corotational geometric transformation is illustrated next. To evaluate the positive semi-definiteness of $\delta\mathbf{K} = \mathbf{K}_\mathbf{i} - \mathbf{K}_\mathbf{t}$, the initial and tangent stiffness matrices of a simple elastic cantilever column element employing the corotational transformation are determined first. The stiffness matrix of the frame element in the local coordinate system is derived as follows.

$$\mathbf{k}_\mathbf{l} = \frac{\partial \mathbf{p}_\mathbf{l}}{\partial \mathbf{u}_\mathbf{l}} = \frac{\partial}{\partial \mathbf{u}_\mathbf{l}}\left(\mathbf{a}^T\mathbf{q}\right) = \mathbf{a}^T\mathbf{k}_\mathbf{b}\,\mathbf{a} + \frac{\partial \mathbf{a}^T}{\partial \mathbf{u}_\mathbf{l}}\mathbf{q} \tag{4.73}$$

where the compatibility matrix $\mathbf{a}$ and the elastic element stiffness matrix $\mathbf{k}_\mathbf{b}$ in the basic coordinate system are given by formulas (4.74). In these formulas $\mathbf{u}_\mathbf{l}$ and $\mathbf{p}_\mathbf{l}$ are the element displacements and forces in the local coordinate system, $\mathbf{u}_\mathbf{b}$ and $\mathbf{q}$ are the element deformations and forces in the basic coordinate system, α is the chord rotation, and L_n is the chord length in the deformed configuration.

$$\mathbf{a} = \begin{bmatrix} -\cos(\alpha) & -\sin(\alpha) & 0 & \cos(\alpha) & \sin(\alpha) & 0 \\ -\frac{\sin(\alpha)}{L_n} & \frac{\cos(\alpha)}{L_n} & 1 & \frac{\sin(\alpha)}{L_n} & -\frac{\cos(\alpha)}{L_n} & 0 \\ -\frac{\sin(\alpha)}{L_n} & \frac{\cos(\alpha)}{L_n} & 0 & \frac{\sin(\alpha)}{L_n} & -\frac{\cos(\alpha)}{L_n} & 1 \end{bmatrix} \tag{4.74}$$

$$\mathbf{k_b} = \begin{bmatrix} \frac{EA}{L} & 0 & 0 \\ 0 & \frac{4EI}{L} & \frac{2EI}{L} \\ 0 & \frac{2EI}{L} & \frac{4EI}{L} \end{bmatrix}$$

Furthermore, the first part of the local stiffness matrix in (4.73) represents the material stiffness and the second part represents the geometric stiffness. These two parts can easily be evaluated with any computer algebra system (CAS) such as Mathematica (Wolfram) or Maple (Maplesoft). However, the resulting 6x6 stiffness matrix gets quite involved and is therefore not shown here. For the elastic cantilever column (see Fig. 4.18a), which is used in this example, the three fixed degrees of freedom at the base do not enter into the equations of motion, and the local initial and tangent stiffness matrices reduce to 3x3 matrices. The tangent stiffness matrix is still quite complicated and is therefore also not shown here, but for the chord rotation $\alpha = 0$ and the chord length $L_n = L$, the following initial stiffness matrix is obtained.

$$\mathbf{k_{1,i}} = \begin{bmatrix} \frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} \tag{4.75}$$

Next the positive semi-definiteness of $\delta\mathbf{K}$ is evaluated by determining its eigenvalues. In order to obtain a closed-form solution of the smallest eigenvalue, the special case of the elastic, corotational truss without loads is considered first ($EI = q_1 = q_2 = q_3 = 0$). The smallest eigenvalue is given by the following expression.

$$\lambda_{\min} = -\frac{EA}{L}\sin(\alpha) \tag{4.76}$$

Since the smallest eigenvalue is negative, $\delta\mathbf{K}$ is no longer positive semi-definite. Thus, the generalized-alpha-OS method is rendered conditionally stable.

Similar results can be obtained numerically if a pushover analysis is performed, where the horizontal displacement $u_{l,5}$ is prescribed at the tip of the elastic cantilever column, which is loaded in compression by a constant vertical force P. As can be seen from Fig. 4.18b, one of the three eigenvalues is negative and decreasing with increasing drift. This again makes $\delta\mathbf{K}$ negative definite and renders the generalized-alpha-OS method conditionally stable. Fig. 4.18c and d show the local forces at the tip of the cantilever and the axial force, respectively. Because the smallest eigenvalue turns negative from the beginning of the pushover analysis, the generalized-alpha-OS method remains conditionally stable even for the analysis of a cantilever column that exhibits a softening material behavior after the initial elastic regime.

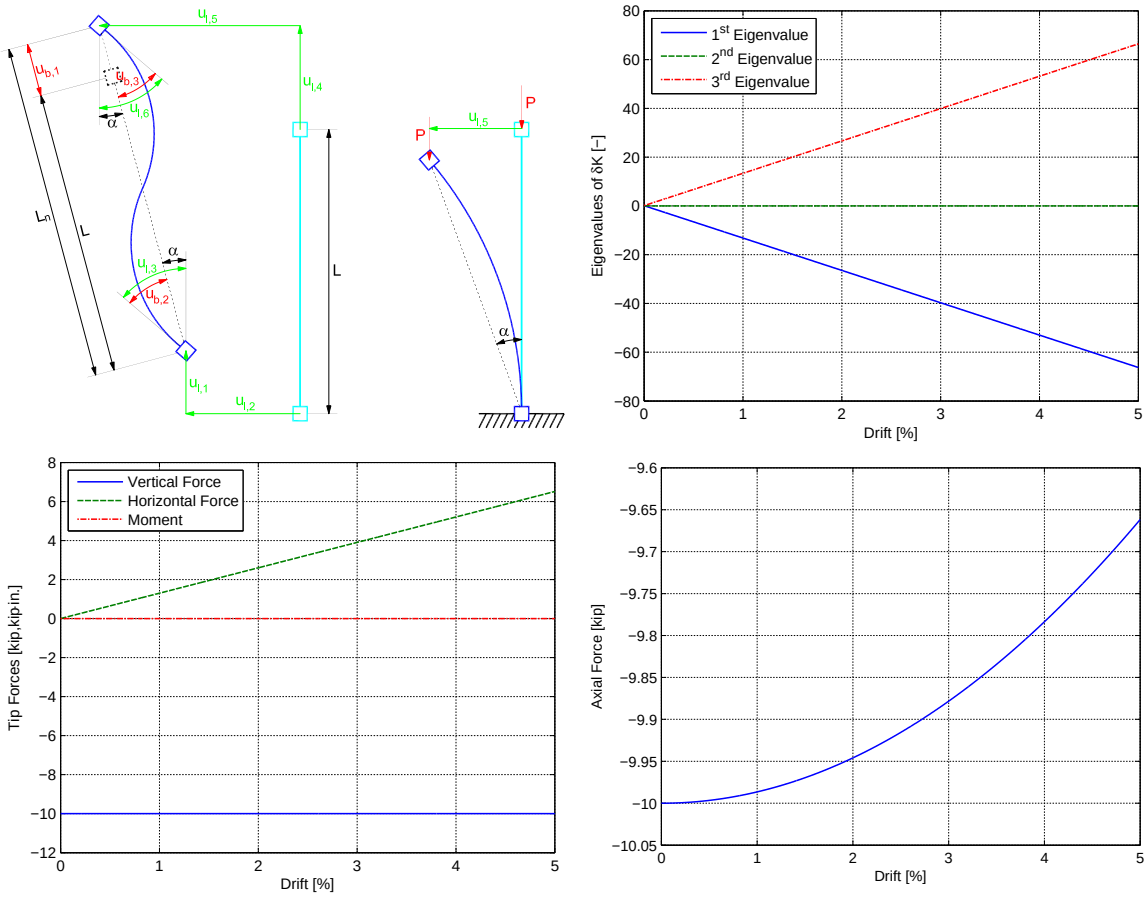


Fig. 4.18 Pushover analysis of elastic cantilever column with corotational geometric transformation: (a) geometry, (b) eigenvalues, (c) tip forces, (d) axial force.

4.4.6 Modified Generalized-Alpha-OS Method (GaOS2)

To improve on the deficiency that the generalized-alpha-OS method can become conditionally stable when analyzing structural models that account for large-displacement geometric nonlinearities, it is suggested to use a mixed stiffness matrix instead of the initial stiffness matrix to approximate the resisting forces. The mixed global stiffness matrix is assembled from the tangent stiffness matrices of the analytical elements and the initial or if available tangent stiffness matrices of the experimental elements. Using the mixed global stiffness matrix, the internal resisting forces on the left-hand side of the equilibrium equations in (4.63) are now approximated as follows.

$$\mathbf{P}_{\mathbf{r},i+\alpha_f} \cong \mathbf{K}_{\mathbf{m}}\,\mathbf{U}_{i+\alpha_f} - \left(\mathbf{K}_{\mathbf{m}}\,\tilde{\mathbf{U}}_{i+\alpha_f} - \tilde{\mathbf{P}}_{\mathbf{r},i+\alpha_f}\right) \tag{4.77}$$

The mixed stiffness matrix is evaluated at the predictor displacements $\tilde{\mathbf{U}}_{i+1}$ or $\tilde{\mathbf{U}}_{i+\alpha_f}$ depending on the employed generalized quadrature rule. This means that the Jacobian is no longer constant and needs to be evaluated once per time step, increasing the computational cost of the algorithm.

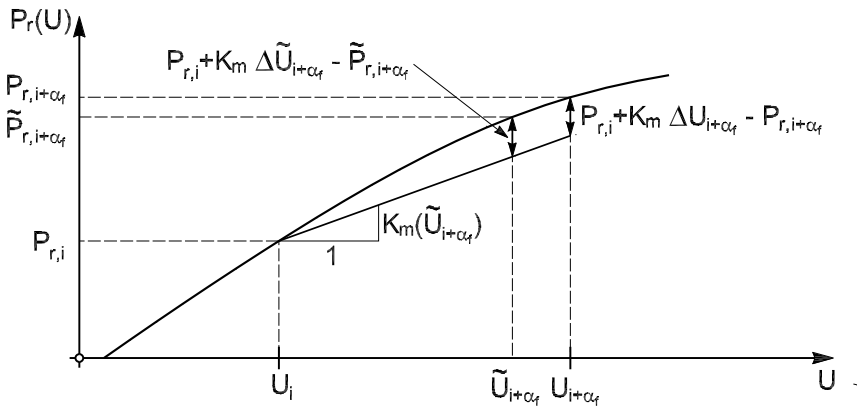


Fig. 4.19 Approximation of nonlinear resisting force using $\mathbf{K_m}$.

Substitution of (4.77) into the balance Equation (4.63) leads to the following approximate equilibrium equations.

$$\mathbf{M}\,\ddot{\mathbf{U}}_{i+\alpha_m} + \mathbf{C}\,\dot{\mathbf{U}}_{i+\alpha_f} + \mathbf{K}_{\mathbf{m}}\left(\mathbf{U}_{i+\alpha_f} - \tilde{\mathbf{U}}_{i+\alpha_f}\right) + \tilde{\mathbf{P}}_{\mathbf{r},i+\alpha_f} = \mathbf{P}_{i+\alpha_f} - \mathbf{P}_{\mathbf{0},i+\alpha_f} \tag{4.78}$$

The rest of the algorithm remains unchanged from the previously described one, and thus, the modified generalized-alpha-OS integration method can be summarized as pseudo-code as shown below in Fig. 4.20.

Initialize:

$\mathbf{U}_0$, $\dot{\mathbf{U}}_0$, $\ddot{\mathbf{U}}_0$

Analyze:

for $(i=0,\ i<N,\ i++)$

$$\tilde{\mathbf{U}}_{i+1} = \mathbf{U}_i + \Delta t\,\dot{\mathbf{U}}_i + \frac{\Delta t^2}{2}(1-2\beta)\,\ddot{\mathbf{U}}_i$$

$$\dot{\tilde{\mathbf{U}}}_{i+1} = \dot{\mathbf{U}}_i + \Delta t(1-\gamma)\,\ddot{\mathbf{U}}_i$$

$$\mathbf{K}_{\mathbf{eff}} = \alpha_m c_3\,\mathbf{M} + \alpha_f c_2\,\mathbf{C} + \alpha_f c_1\,\mathbf{K}_{\mathbf{m}}$$

$$\mathbf{P}_{\mathbf{eff}} = \mathbf{P}_{i+\alpha_f} - \mathbf{P}_{\mathbf{0},i+\alpha_f} - \tilde{\mathbf{P}}_{\mathbf{r},i+\alpha_f} \qquad \textit{(Eq. 4.67 or Eq. 4.68)}$$

$$-\mathbf{K}_{\mathbf{m}}(1-\alpha_f)(\mathbf{U}_i - \tilde{\mathbf{U}}_i) - \mathbf{M}(1-\alpha_m)\ddot{\mathbf{U}}_i - \mathbf{C}\left[(1-\alpha_f)\dot{\mathbf{U}}_i + \alpha_f\dot{\tilde{\mathbf{U}}}_{i+1}\right]$$

$$\Delta\mathbf{U} = solve(\mathbf{K}_{\mathbf{eff}}\,\Delta\mathbf{U} = \mathbf{P}_{\mathbf{eff}})$$

$$\mathbf{U}_{i+1} = \tilde{\mathbf{U}}_{i+1} + c_1\,\Delta\mathbf{U}$$

$$\dot{\mathbf{U}}_{i+1} = \dot{\tilde{\mathbf{U}}}_{i+1} + c_2\,\Delta\mathbf{U}$$

$$\ddot{\mathbf{U}}_{i+1} = c_3\,\Delta\mathbf{U}$$

end

Fig. 4.20 Modified generalized-alpha-OS (GαOS2) method.

To demonstrate the improved stability of the modified generalized-alpha-OS method, a cantilever column is analyzed using the large-displacement, corotational geometric transformation. As can be seen from Fig. 4.21a, the model consists of two force-beam-column elements of equal length. Since mass is assigned only to the two translational degrees of freedom at the top of the column, the global mass matrix of the model is singular with rank of two. The fundamental period of the cantilever column is 1.321 sec and damping is assumed to be 5% of critical. Before the model is subjected to the 1978 Tabas near-fault ground motion scaled to a pga of $0.378\,g$, the cantilever column is loaded in compression by a constant vertical gravity load of $P = 5\,\text{kip}$. The structure is analyzed using an integration time-step size of $\Delta t_{\text{int}} = 0.01\,\text{sec}$, which is equal to the time increment of the ground acceleration record.

As can be seen from Fig. 4.21b and Fig. 4.22b, the displacement response of the cantilever column, analyzed by the generalized-alpha-OS method, starts growing without bounds at approximately 6.5 sec into the simulation. Therefore, it becomes unstable. On the other hand, the displacement response of the cantilever column, analyzed by the modified generalized-alpha-OS method, remains bounded and accurate for the entire ground motion analysis. This is due to

the improved stability characteristics of the algorithm, which is achieved by using a mixed global stiffness matrix. By comparing Figures 4.21 and 4.22, it can also be observed that the same general behavior is attained for both the cantilever column with the linear-elastic material behavior and the one with the nonlinear material behavior.

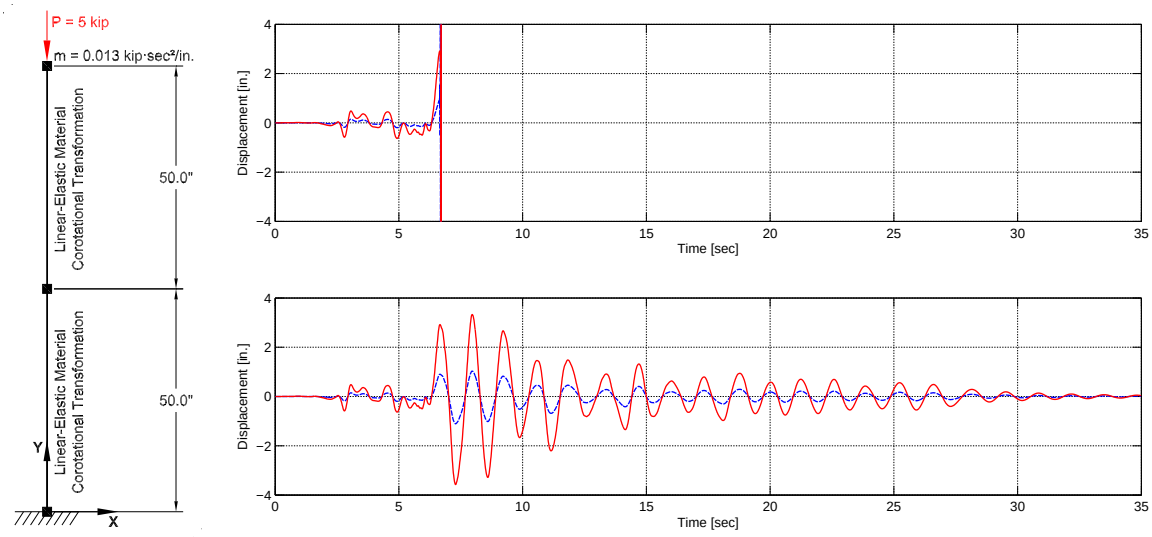


Fig. 4.21 Cantilever column response for linear-elastic material behavior and nonlinear, corotational transformation: (a) model, (b) GαOS1, (c) GαOS2.

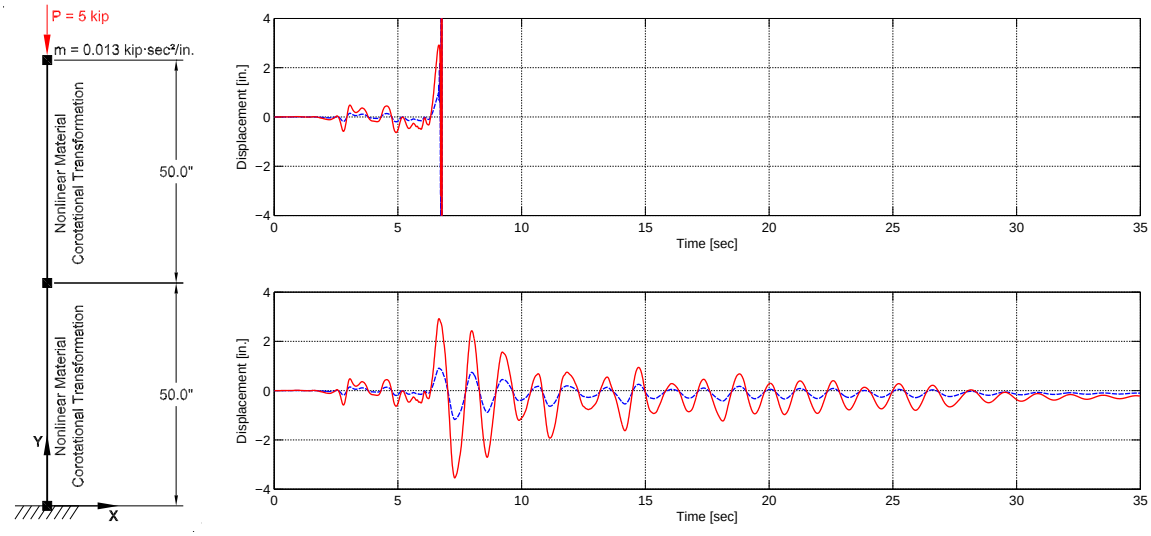


Fig. 4.22 Cantilever column response for nonlinear material behavior and nonlinear, corotational transformation: (a) model, (b) GαOS1, (c) GαOS2.

4.4.7 Accuracy: Numerical Dissipation and Dispersion

In structural dynamics, it is often more useful to introduce different accuracy measures than local truncation error. Accuracy measures, in terms of numerical dissipation and dispersion, are preferred. According to Hughes (2000), for a linear-elastic, damped single-degree-of-freedom system in free vibration, the discrete solution of the numerical methods can be put into a form similar to the exact solution.

$$u(t) = e^{-\bar{\xi}\bar{\omega}_n t}\left(A\cos(\bar{\omega}_D t) + B\sin(\bar{\omega}_D t)\right) \tag{4.79}$$

where $\bar{\omega}_D = \bar{\omega}_n\sqrt{1-\bar{\xi}^2}$. In the above formula, $\bar{\xi}$ and $\bar{\omega}_n$ are the algorithmic counterparts of the damping ratio and the circular frequency of the exact solution, respectively. $\bar{\xi}$ is called the algorithmic or numerical damping ratio (a measure for dissipation) and $(\bar{T}-T)/T$ is referred to as the relative period error (a measure for dispersion). Since it is difficult to obtain analytical expressions of these accuracy measures for most numerical schemes, they are obtained from simulation results instead. Thus, free vibration tests of a linear-elastic single-degree-of-freedom system are performed for all the direct integration methods in order to determine their numerical dissipation and dispersion properties. The SDOF system has a period of $T = 1\text{sec}$ and is excited by an initial displacement of $\mathbf{U}_0 = 1$ in. and an initial velocity of $\dot{\mathbf{U}}_0 = 0$ in./sec.

Figures 4.23–4.26 present the displacement responses of the free vibration solutions for one specific normalized integration time step of $\Delta t/T = 0.1$. The first figure in this series provides a comparison of the displacements for the discussed direct integration methods. As can be seen, the explicit Newmark method and the explicit generalized-alpha methods shorten periods, whereas all the implicit methods including the operator-splitting methods lengthen periods. Furthermore, it can be observed how the class of generalized-alpha methods provides the means of introducing user-adjustable, algorithmic energy dissipation by specifying the spectral radius ρ.

Fig. 4.24 shows the free vibration solutions generated by the explicit generalized-alpha method for a range of spectral radii ρ_b at the bifurcation frequency. For $\rho_b = 1.0$, no algorithmic damping is introduced and the solution is identical to the one obtained by the explicit Newmark method. On the other hand, for $\rho_b = 0.0$, the integration scheme achieves asymptotic annihilation, providing maximal algorithmic damping. This means that depending on the chosen integration time step, any high-frequency response is almost annihilated in a single time step.

This high-frequency dissipation of the explicit generalized-alpha method is very useful when analyzing structures where constraints are enforced by the penalty method.

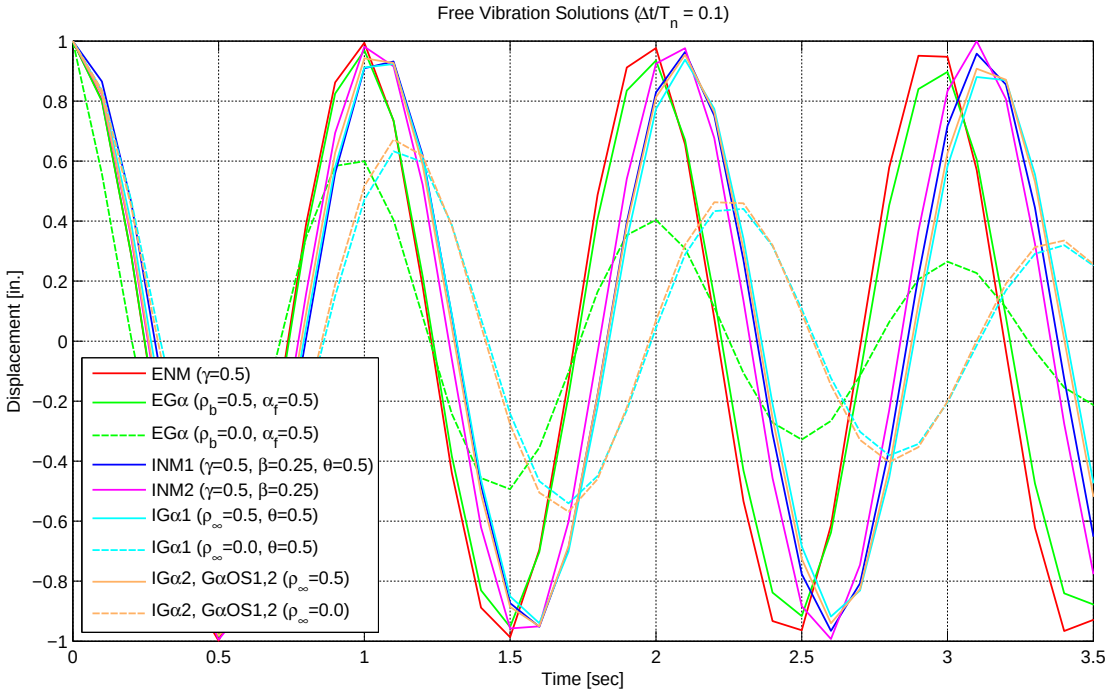


Fig. 4.23 Free vibration solutions for direct integration methods.

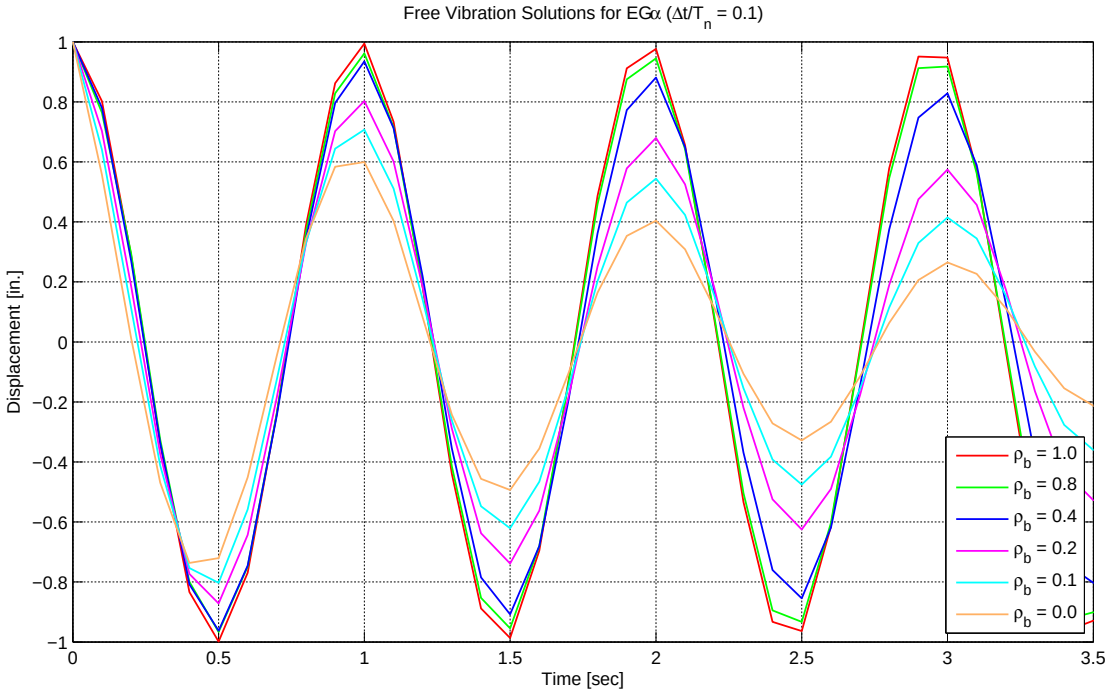


Fig. 4.24 Free vibration solutions for explicit generalized-alpha (EGα) method.

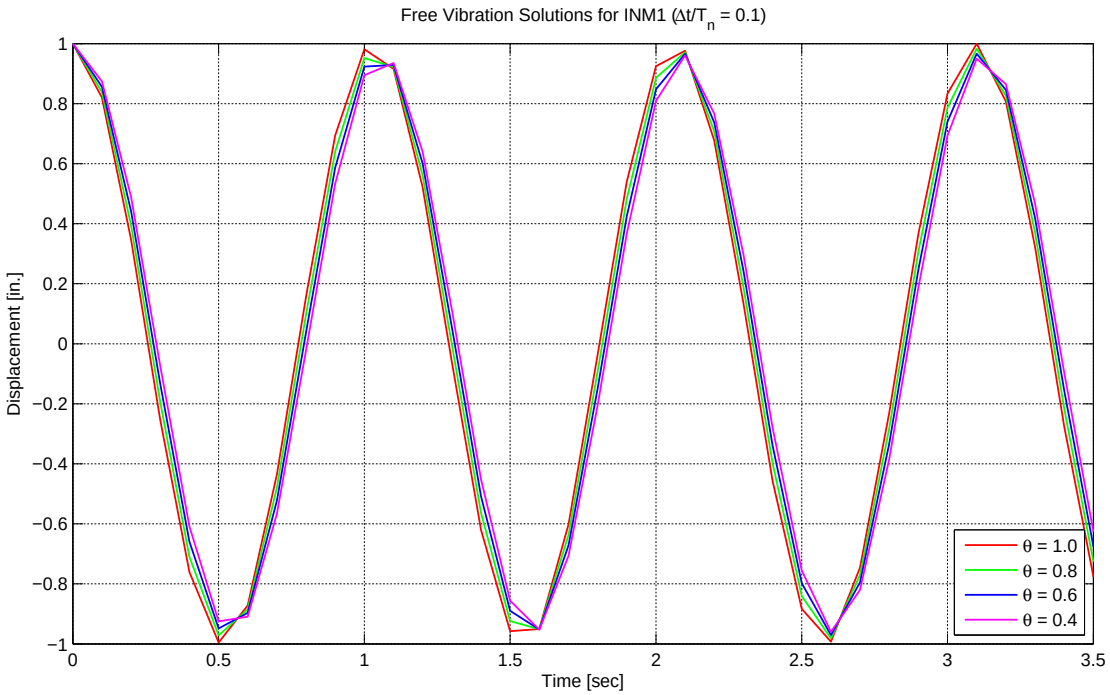


Fig. 4.25 Free vibration solutions for implicit Newmark (INM1) method.

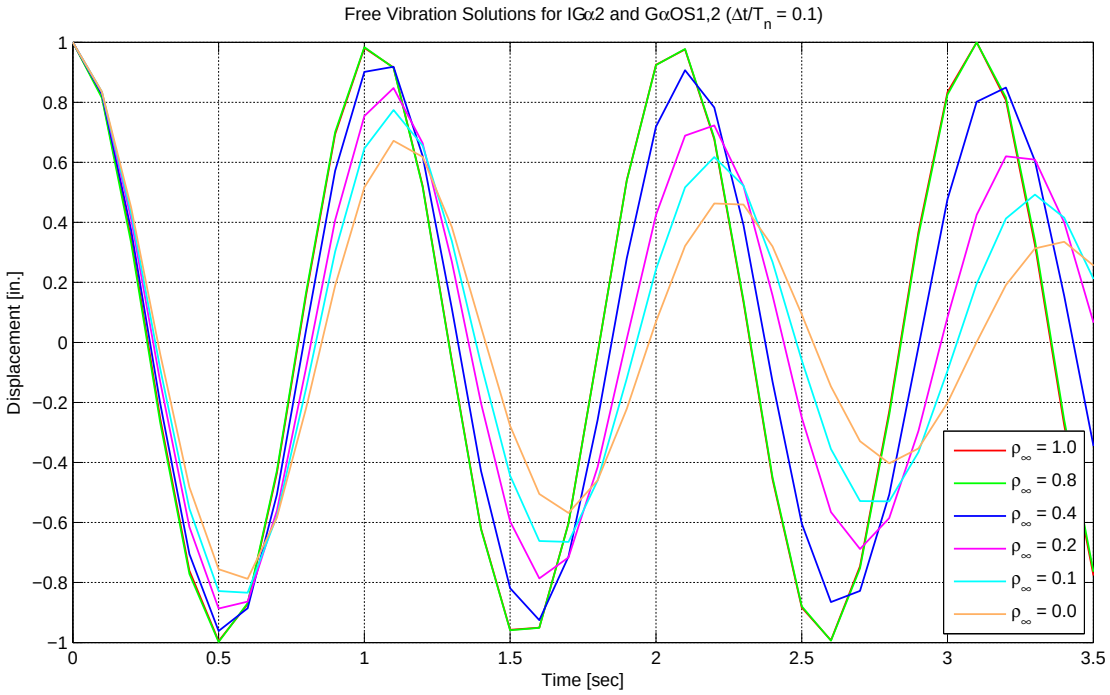


Fig. 4.26 Free vibration solutions for implicit generalized-alpha (IGα2) method.

The displacement responses of the modified implicit Newmark method for different values of the increment reduction factor θ are shown in Fig. 4.25. It can be seen that the free vibration solutions lose some accuracy with the reduction of the increment reduction factors.

Fig. 4.26, the last in this series, presents the displacement responses for the constant number of iterations implicit generalized-alpha method and the generalized-alpha-OS methods, which are identical for linear-elastic systems. Similar to the explicit method, no algorithmic damping is introduced when $\rho_\infty = 1.0$ and the two integration schemes produce equivalent solutions to the implicit Newmark method. For $\rho_\infty = 0.0$, the algorithms again reach asymptotic annihilation, thereby introducing maximal algorithmic damping in the high-frequency modes while minimizing energy dissipation in the low-frequency modes. As the explicit generalized-alpha method, these two integration schemes are thus well suited when constraints are enforced by the penalty method.

The next three figures present comparisons of the two accuracy measures, that is, the algorithmic damping ratios and the relative period errors, for all the discussed direct integration methods. Fig. 4.27 provides a comparison among the different schemes.

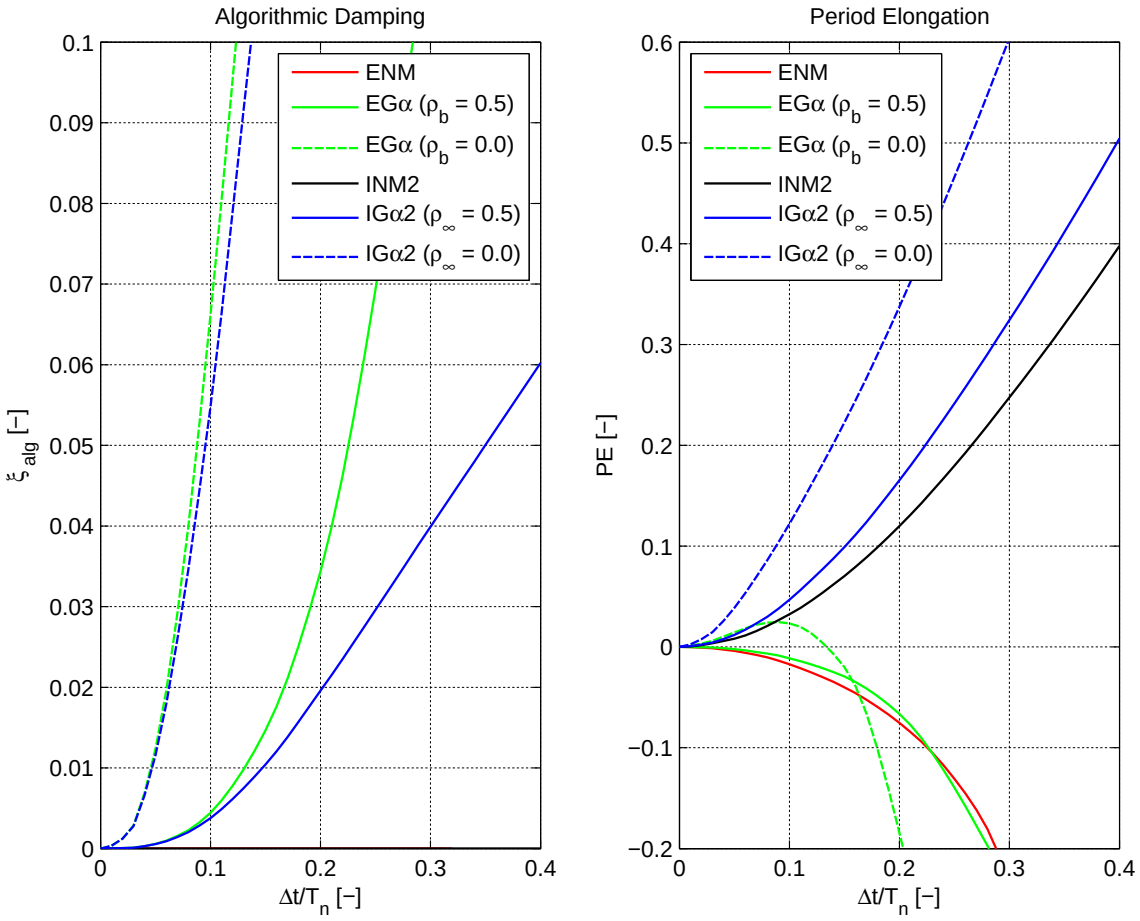


Fig. 4.27 Algorithmic damping ratios and relative period errors for direct integration methods.

It can be seen that both the explicit and the implicit Newmark methods provide zero algorithmic energy dissipation in all the modes. In contrast, the generalized-alpha methods provide user-adjustable amounts of algorithmic damping for spectral radii $\rho < 1.0$. Furthermore, since the explicit generalized-alpha method is conditionally stable, the numerical damping ratios increase more rapidly as the stability limits are approached, than for the implicit generalized-alpha and the generalized-alpha-OS schemes. As can be seen from the second graph in Fig. 4.27, all the algorithms exhibit some relative period error. The explicit Newmark method and the explicit generalized-alpha methods generally shorten the periods, and the implicit methods elongate the periods.

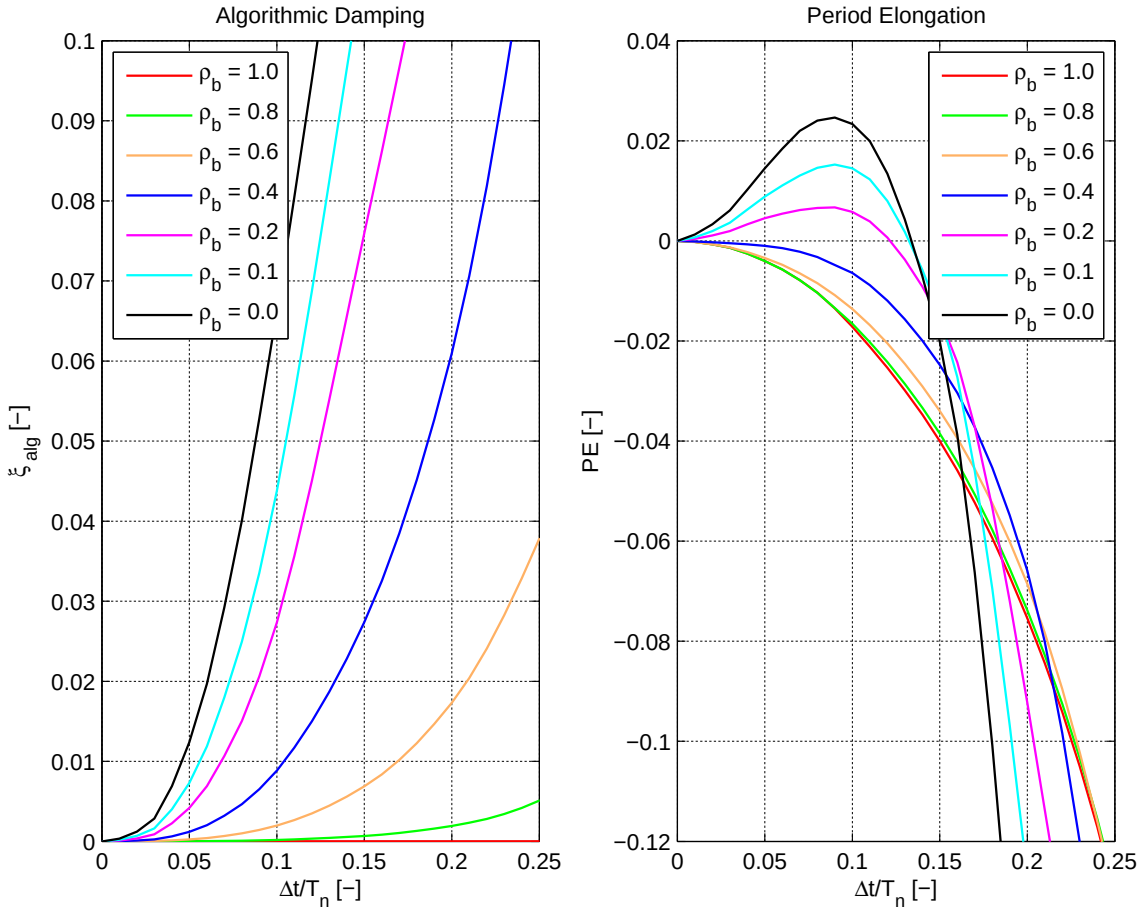


Fig. 4.28 Algorithmic damping ratios and relative period errors for explicit generalized-alpha (EGα) method.

Fig. 4.28 above, presents the numerical damping ratios and the relative period errors of the explicit generalized-alpha method for different values of the spectral radius ρ_b. The

algorithmic energy dissipation increases from zero when $\rho_b = 1.0$ to the maximal value when $\rho_b = 0.0$. As can be seen from the second part of Fig. 4.28, both period shortening and elongation can occur, depending on the spectral radius ρ_b. As pointed out by Hulbert and Chung (1996), period errors are minimized in the low-frequency domain at an approximate spectral radius of $\rho_b \cong 0.3665$.

Finally, for the implicit generalized-alpha and the generalized-alpha-OS methods, Fig. 4.29 shows the algorithmic damping ratios and the relative period errors versus the normalized integration time step, for a range of spectral radii ρ_∞. As with the related explicit method, the algorithmic energy dissipation increases from zero when $\rho_\infty = 1.0$ to the maximal value when $\rho_\infty = 0.0$. On the other hand, the integration schemes elongate periods for all values of the spectral radius ρ_∞. The most accurate response is obtained for $\rho_\infty = 1.0$, which is equivalent to the implicit Newmark method, but does not provide any algorithmic energy dissipation.

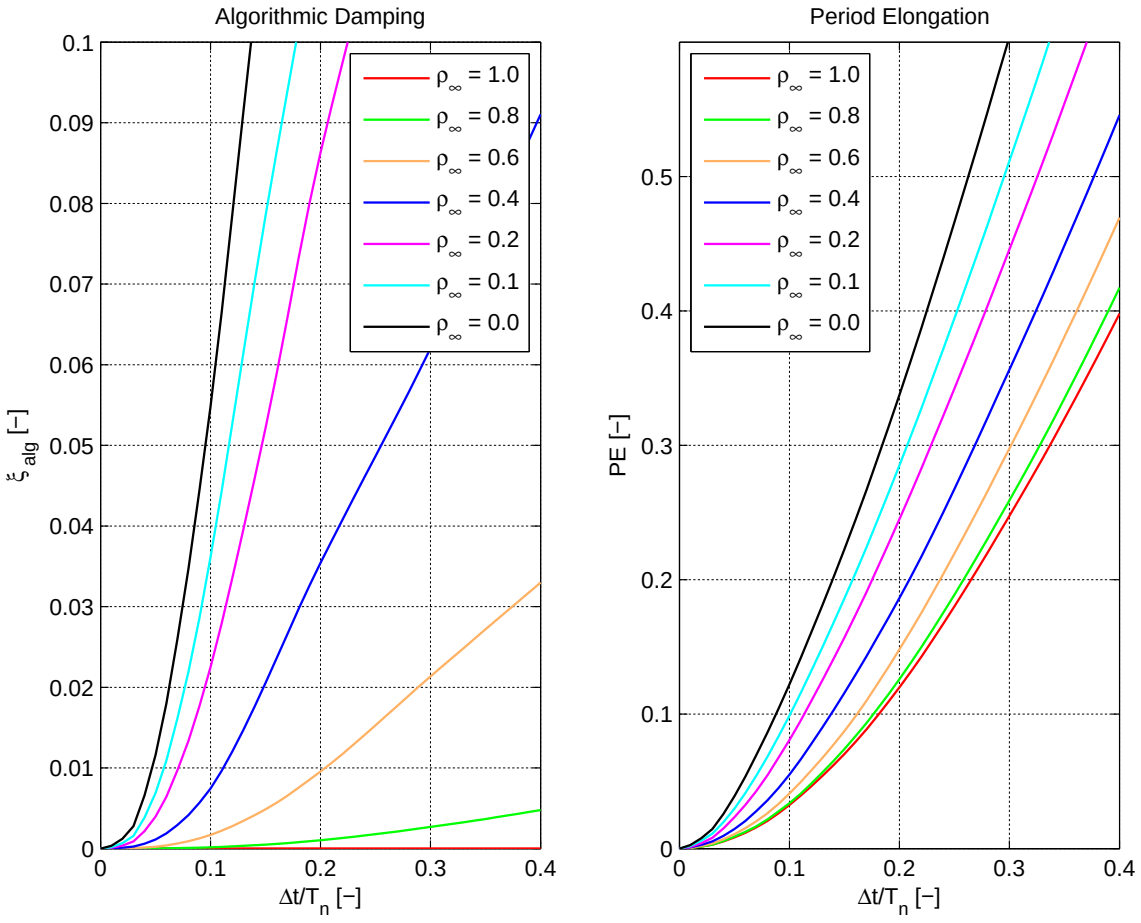


Fig. 4.29 Algorithmic damping ratios and relative period errors for implicit generalized-alpha (IGα) and generalized-alpha-OS (GαOS) methods.

All of the direct integration methods that have been presented above and that are summarized in the next subsection have been implemented in the Open System for Earthquake Engineering Simulation, OpenSees (McKenna 1997), finite element software and are available to users to solve the equations of motion when performing hybrid simulations.

4.4.8 Summary of Special Requirements

Table 4.3 Summary of special requirements for direct integration methods.

Requirement	*ENM*	*EGα*	*INM1*	*INM2*	*IGα1*	*IGα2*	*GαOS1*	*GαOS2*
1. $p \geq 2$ accurate	$p=2$	$p=2$	$p=2$	$p=2$	$p=2$ (1)	$p=2$ (1)	$p=2$ (1)	$p=2$ (1)
2. few function calls per Δt	1	1		k_{max}		k_{max}	1	1
3. unconditionally stable			✓	✓	✓	✓	✓(2)	✓(3)
4. constant $\mathbf{k}_{el,exp}$	na	na	✓	✓	✓	✓	✓	✓
5. algorithmic damping		✓			✓	✓	✓	✓
6. strictly incr./decreasing $\Delta\mathbf{U}$	na	na	✓	✓	✓	✓	na	na
7. uniform $\Delta\mathbf{U}$	na	na		✓		✓	na	na
8. real-time compatible	✓	✓		✓		✓	✓	✓

Notes: (1) acceleration is $p=1$ accurate

(2) only if $\delta\mathbf{K} = \mathbf{K}_i - \mathbf{K}_t$ is positive semi-definite

(3) only if $\delta\mathbf{K} = \mathbf{K}_m - \mathbf{K}_t$ is positive semi-definite

4.5 RUNGE-KUTTA METHODS

Runge-Kutta methods belong to the class of one-step methods. This implies that only the current solution state at time t_i is required to calculate the new solution at $t_i + \Delta t$. As mentioned earlier, these solution schemes are designed to solve first-order ordinary differential equations, which require that higher-order differential equations, like the equations of motion, need to be transformed into a system of first-order differential equations before the Runge-Kutta methods can be applied. They are essentially always stable and can be either explicit or implicit. The objective of Runge-Kutta methods is to build a series of *s stages* approximating the vector function $\mathbf{f}(t, \mathbf{y})$ between t_i and t_{i+1}, and then use a linear combination of those stages to advance the numerical solution to the new time step t_{i+1}.

The system of first-order ordinary differential equations to which the Runge-Kutta methods are applied is an initial value problem that has the following form.

$$\begin{aligned} \dot{\mathbf{y}}(t) &= \mathbf{f}(t, \mathbf{y}) \\ \mathbf{y}(0) &= \mathbf{y_0} \end{aligned} \tag{4.80}$$

After Runge (1895) and Heun (1900) constructed numerical schemes based on the Euler method, it was Kutta (1901) who formulated the generalized scheme of what is now known as the Runge-Kutta method. This method has *s stages* and is expressed in the following manner.

$$\begin{aligned} \mathbf{k}_m &= \mathbf{f}\left(t_{i+c_m},\, \mathbf{y}_i + \Delta t \sum_{n=1}^{s} a_{mn} \mathbf{k}_n \right) \quad \text{for m=1..s} \\ \mathbf{y}_{i+1} &= \mathbf{y}_i + \Delta t \sum_{n=1}^{s} b_n \mathbf{k}_n \end{aligned} \tag{4.81}$$

An alternative form for fully implicit Runge-Kutta equations, which is numerically advantageous for implementation, takes the following form.

$$\begin{aligned} \mathbf{z}_m &= \Delta t \sum_{n=1}^{s} a_{mn} \mathbf{f}\left(t_{i+c_n},\, \mathbf{y}_i + \mathbf{z}_n \right) \quad \text{for m=1..s} \\ \mathbf{y}_{i+1} &= \mathbf{y}_i + \sum_{n=1}^{s} \hat{b}_n \mathbf{z}_n \end{aligned} \tag{4.82}$$

where $\mathbf{A} = [a_{mn}]$ and $\hat{\mathbf{b}}^T = \mathbf{b}^T \ \mathbf{A}^{-1}$. Because $\mathbf{A}$ needs to be nonsingular for inversion, this form of the equations is possible only for fully implicit methods. Finally, it is customary to symbolize Runge-Kutta methods by the Butcher array as follows.

$$\begin{array}{c|c} \mathbf{c} & \mathbf{A} \\ \hline & \mathbf{b}^T \end{array} \tag{4.83}$$

As explained earlier the second-order equations of motion have to be transformed into a system of first-order differential equations, which is repeated here for convenience.

$$\begin{aligned} \dot{\mathbf{y}}(t) &= \mathbf{f}(t,\mathbf{y}) = \begin{Bmatrix} \mathbf{y}_2 \\ \mathbf{M}^{-1}\left(\mathbf{P}(t) - \mathbf{P}_0(t) - \mathbf{C}\mathbf{y}_2 - \mathbf{P}_r(\mathbf{y}_1)\right) \end{Bmatrix} \\ \mathbf{y}(0) &= \begin{Bmatrix} \mathbf{U}_0 \\ \dot{\mathbf{U}}_0 \end{Bmatrix} \end{aligned} \tag{4.84}$$

where $\mathbf{y} = \left\{\mathbf{U}(t), \dot{\mathbf{U}}(t)\right\}^T$ is a stacked vector containing the structural displacements and velocities.

4.5.1 Explicit Methods (ERK)

For explicit Runge-Kutta methods, $\mathbf{A}$ is strictly lower triangular ($a_{mn} = 0 \quad \forall n \geq m$) and Equation (4.84) is simply evaluated at different arguments. Unfortunately, all of these explicit algorithms need to invert the mass matrix $\mathbf{M}$ so that they cannot be used to analyze structural problems with singular mass matrices. In addition, none of the ERK methods are A-stable. Several popular ERK methods including their Butcher arrays are summarized next.

Runge (ERK2):

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}\left(t_i, \mathbf{y}_i\right) \\ \mathbf{k}_2 &= \mathbf{f}\left(t_{i+1/2}, \mathbf{y}_i + \frac{1}{2}\Delta t\, \mathbf{k}_1\right) \\ \mathbf{y}_{i+1} &= \mathbf{y}_i + \Delta t\, \mathbf{k}_2 \end{aligned} \qquad \begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array} \tag{4.85}$$

The explicit Runge method is consistent and accurate of order $p = 2$, satisfying requirement 1. Since this scheme has two stages, it requires two function calls per time step. This means that one additional function call per time step is needed as compared to any of the previously discussed second-order explicit direct integration methods.

Heun (ERK3):

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}\left(t_i,\, \mathbf{y}_i\right) \\
\mathbf{k}_2 &= \mathbf{f}\left(t_{i+1/3},\, \mathbf{y}_i + \frac{1}{3}\Delta t\, \mathbf{k}_1\right) \\
\mathbf{k}_3 &= \mathbf{f}\left(t_{i+2/3},\, \mathbf{y}_i + \frac{2}{3}\Delta t\, \mathbf{k}_2\right) \\
\mathbf{y}_{i+1} &= \mathbf{y}_i + \Delta t\left(\frac{1}{4}\mathbf{k}_1 + \frac{3}{4}\mathbf{k}_3\right)
\end{aligned} \qquad \begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \frac{2}{3} & 0 & \frac{2}{3} & 0 \\ \hline & \frac{1}{4} & 0 & \frac{3}{4} \end{array} \tag{4.86}$$

The explicit Heun method is consistent and accurate of order $p = 3$, and thus, satisfies requirement 1. The method requires three function calls per time step. Since it is more accurate than any of the direct integration methods, it is an excellent alternative.

Runge-Kutta 3/8 Rule (ERK4b):

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}\left(t_i,\, \mathbf{y}_i\right) \\
\mathbf{k}_2 &= \mathbf{f}\left(t_{i+1/3},\, \mathbf{y}_i + \frac{1}{3}\Delta t\, \mathbf{k}_1\right) \\
\mathbf{k}_3 &= \mathbf{f}\left(t_{i+2/3},\, \mathbf{y}_i + \Delta t\left(-\frac{1}{3}\mathbf{k}_1 + \mathbf{k}_2\right)\right) \\
\mathbf{k}_4 &= \mathbf{f}\left(t_{i+1},\, \mathbf{y}_i + \Delta t\left(\mathbf{k}_1 - \mathbf{k}_2 + \mathbf{k}_3\right)\right) \\
\mathbf{y}_{i+1} &= \mathbf{y}_i + \Delta t\left(\frac{1}{8}\mathbf{k}_1 + \frac{3}{8}\mathbf{k}_2 + \frac{3}{8}\mathbf{k}_3 + \frac{1}{8}\mathbf{k}_4\right)
\end{aligned} \qquad \begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\ \frac{2}{3} & -\frac{1}{3} & 1 & 0 & 0 \\ 1 & 1 & -1 & 1 & 0 \\ \hline & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{array} \tag{4.87}$$

The explicit 3/8 Rule is consistent and accurate of order $p = 4$, satisfying requirement 1. According to the four stages, the method also requires four function calls per time step. However, the last stage determines an approximation for the derivative at the end of the current time step, and the first stage determines an approximation for the derivative at the beginning of the new time step. This means that the resisting forces are acquired twice at time t_{i+1}, which is not ideal for generating strictly increasing or decreasing and uniform displacement increments as stated by requirements 6 and 7.

To investigate the continuity and uniformity of the displacement command signals that are imposed on the experimental parts of a structure, a two-degrees-of-freedom, one-bay-frame model is analyzed by the different explicit Runge-Kutta integration methods. The one-bay-frame model is equivalent to the one that was used for the rapid geographically distributed hybrid

simulations (see Chapter 6). It is defined by two column elements that are connected by a fairly flexible linear-elastic truss element. The left nonlinear column is tested experimentally, while the right linear-elastic column is modeled analytically. Only material nonlinearities are considered in this model, meaning that geometric nonlinearities due to axial forces are ignored. Fig. 4.30 shows close-ups of the displacement command signals at the controlled degree of freedom of the experimental element over three integration time steps for the different ERK methods.

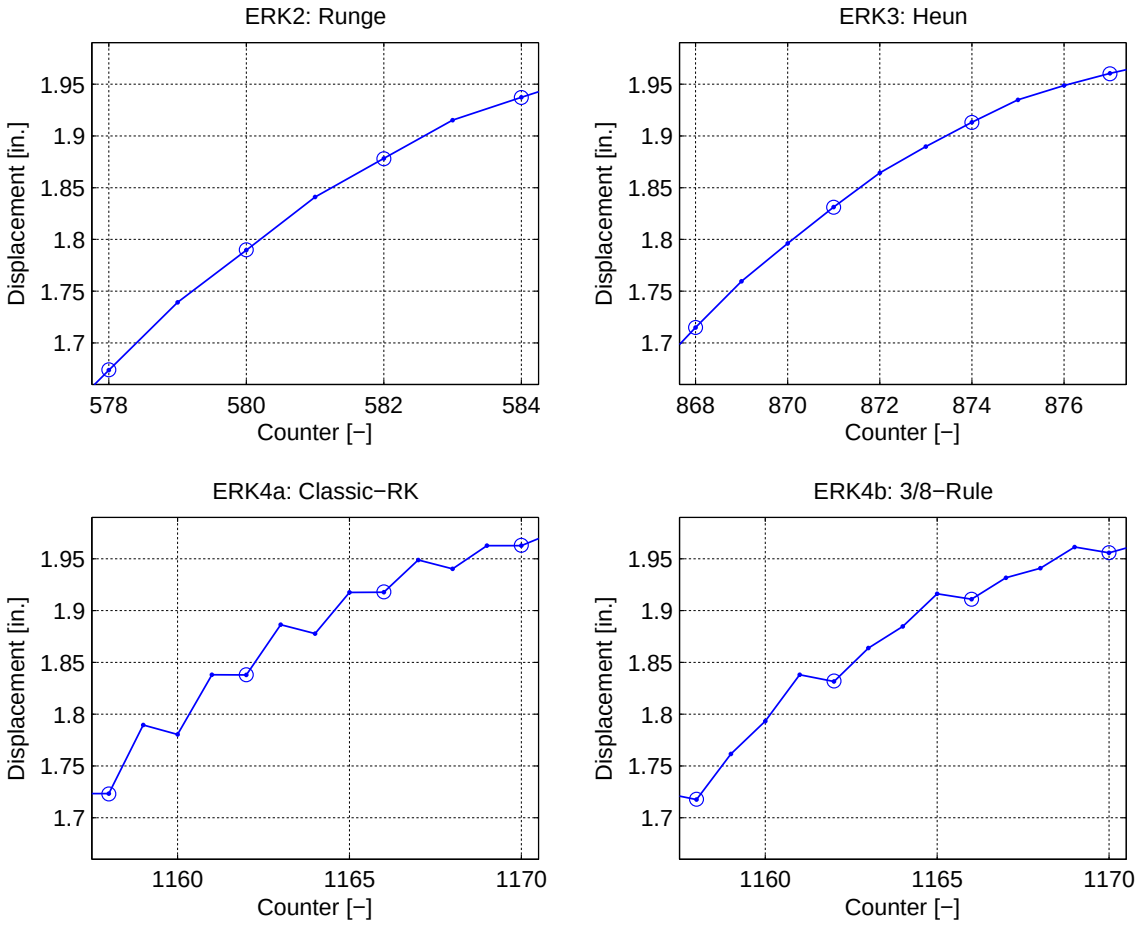


Fig. 4.30 ERK convergence comparison for one-bay-frame model.

As can be seen from the plots, the second-order Runge method and the third-order Heun method produce very smooth and uniform displacement commands. On the other hand, the command displacements of both fourth-order methods are neither strictly increasing nor uniform. According to the time increment vector $\mathbf{c}^T = \{0,\ 1/2,\ 1/2,\ 1\}$, the classic-RK method generates double displacement commands twice at $t_{i+1/2}$ and t_{i+1}, while the 3/8-rule method with $\mathbf{c}^T = \{0,\ 1/3,\ 2/3,\ 1\}$ does so only once at t_{i+1}. Due to the order conditions, any explicit Runge-

Kutta scheme of order $p \geq 4$ requires the two stages at t_i and t_{i+1}. This means that displacement commands are inevitably generated and sent to the experimental portions of the structure twice at the same time instance t_{i+1}. Because of this, these higher-order explicit Runge-Kutta methods should not be used for hybrid simulation. However, for the analysis of purely analytical structures such methods are compelling numerical integration schemes due to their improved accuracy.

Finally, the preferred explicit Runge-Kutta method for hybrid simulation is summarized as pseudo-code in Fig. 4.31. The implementation simply consists of the evaluation of the three stages according to Equations (4.86), for each time step.

Initialize:

$\mathbf{U}_0,\ \dot{\mathbf{U}}_0,\ \ddot{\mathbf{U}}_0$

$\mathbf{y}_0 = \left\{\mathbf{U}_0,\ \dot{\mathbf{U}}_0\right\}^T$

Analyze:

for $(i = 0,\ i < N,\ i{+}{+})$

$\mathbf{y}_{i+1} = \mathbf{y}_i$ $\qquad \mathbf{k}_1 = \mathbf{f}\left(t_i,\ \mathbf{y}_{i+1}\right)$ $\qquad$ *(1st stage)*

$\mathbf{y}_{i+1} = \mathbf{y}_i + \frac{1}{3}\Delta t\,\mathbf{k}_1$ $\qquad \mathbf{k}_2 = \mathbf{f}\left(t_{i+1/3},\ \mathbf{y}_{i+1}\right)$ $\qquad$ *(2nd stage)*

$\mathbf{y}_{i+1} = \mathbf{y}_i + \frac{2}{3}\Delta t\,\mathbf{k}_2$ $\qquad \mathbf{k}_3 = \mathbf{f}\left(t_{i+2/3},\ \mathbf{y}_{i+1}\right)$ $\qquad$ *(3rd stage)*

$\Delta\mathbf{y} = \frac{1}{4}\Delta t\left(\mathbf{k}_1 + 3\mathbf{k}_3\right)$

$\mathbf{U}_{i+1} = \mathbf{U}_i + \Delta\mathbf{y_1}$

$\dot{\mathbf{U}}_{i+1} = \dot{\mathbf{U}}_i + \Delta\mathbf{y_2}$

$\ddot{\mathbf{U}}_{i+1} = \ddot{\mathbf{U}}_i + \frac{\Delta\mathbf{y_2}}{\Delta t}$

end

Fig. 4.31 Explicit third-order Heun (ERK3) method.

4.5.2 Implicit Methods (IRK)

Since the mass matrix is often singular in structural dynamics problems, meaning that explicit Runge-Kutta methods cannot be applied, implicit Runge-Kutta methods that can circumvent this problem are discussed next. First several IRK methods are summarized including their Butcher arrays. Afterwards, the derivation of the implicit Runge-Kutta methods applied to the equations

of motion is explained in terms of the 2-stage Gauss method. All the other lower-order implicit methods follow the same derivation.

Midpoint Rule (IRK2a):

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}\left(t_{i+1/2},\, \mathbf{y}_i + \frac{1}{2}\Delta t\, \mathbf{k}_1 \right) \\ \mathbf{y}_{i+1} &= \mathbf{y}_i + \Delta t\, \mathbf{k}_1 \end{aligned} \qquad \begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array} \tag{4.88}$$

The implicit midpoint rule is consistent and accurate of order $p = 2$, and thus, satisfies requirement 1. Since this method consists of only one stage, the number of function evaluations per time step is equal to the number of iterations necessary to achieve convergence.

Trapezoidal Rule (IRK2b):

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}\left(t_i,\, \mathbf{y}_i \right) \\ \mathbf{k}_2 &= \mathbf{f}\left(t_{i+1},\, \mathbf{y}_i + \Delta t\left(\frac{1}{2}\mathbf{k}_1 + \frac{1}{2}\mathbf{k}_2 \right) \right) \\ \mathbf{y}_{i+1} &= \mathbf{y}_i + \Delta t\left(\frac{1}{2}\mathbf{k}_1 + \frac{1}{2}\mathbf{k}_2 \right) \end{aligned} \qquad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \tag{4.89}$$

Like the implicit midpoint rule, the implicit trapezoidal rule is also consistent and accurate of order $p = 2$. It therefore satisfies requirement 1. However it consists of two stages instead of one, which means that it requires twice as many function evaluations per time step as the implicit midpoint method. It should be noticed that with some algebraic manipulation it is possible to show that the implicit trapezoidal rule is equivalent to the implicit Constant-Acceleration Newmark method.

$$\dot{\mathbf{y}}_{i+1} = \mathbf{f}\left(t_{i+1}, \mathbf{y}_{i+1} \right) = \left\{ \begin{array}{c} \dot{\mathbf{U}}_{i+1} \\ \mathbf{M}^{-1}\left(\mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} - \mathbf{C}\,\dot{\mathbf{U}}_{i+1} - \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1} \right) \right) \end{array} \right\} \tag{4.90}$$

From the second formula in (4.90), the equilibrium equations at time t_{i+1} are recovered.

$$\mathbf{M}\,\ddot{\mathbf{U}}_{i+1} + \mathbf{C}\,\dot{\mathbf{U}}_{i+1} + \mathbf{P}_{\mathbf{r}}\left(\mathbf{U}_{i+1} \right) = \mathbf{P}_{i+1} - \mathbf{P}_{\mathbf{0},i+1} \tag{4.91}$$

Because the second stage is equal to $\mathbf{k}_2 = \mathbf{f}\left(t_{i+1},\, \mathbf{y}_{i+1} \right)$, substitution of $\dot{\mathbf{y}} = \mathbf{f}$ into the implicit trapezoidal rule (4.89) yields the following expression.

$$\mathbf{y}_{i+1} = \begin{Bmatrix} \mathbf{U}_{i+1} \\ \dot{\mathbf{U}}_{i+1} \end{Bmatrix} = \begin{Bmatrix} \mathbf{U}_{i} \\ \dot{\mathbf{U}}_{i} \end{Bmatrix} + \Delta t \left(\frac{1}{2} \begin{Bmatrix} \dot{\mathbf{U}}_{i} \\ \ddot{\mathbf{U}}_{i} \end{Bmatrix} + \frac{1}{2} \begin{Bmatrix} \dot{\mathbf{U}}_{i+1} \\ \ddot{\mathbf{U}}_{i+1} \end{Bmatrix} \right) \tag{4.92}$$

These two equations are obviously equivalent to Newmark's finite difference formulas (4.13) with $\beta = 1/4$ and $\gamma = 1/2$.

2-Stage Radau IA (IRK3):

The Radau IA methods are based on the left Radau quadrature formulas. This makes these algorithms order $2s-1$ accurate. The 2-stage scheme is defined as follows.

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}\left(t_i,\, \mathbf{y}_i + \Delta t \left(\frac{1}{4}\mathbf{k}_1 - \frac{1}{4}\mathbf{k}_2 \right) \right) \\ \mathbf{k}_2 &= \mathbf{f}\left(t_{i+2/3},\, \mathbf{y}_i + \Delta t \left(\frac{1}{4}\mathbf{k}_1 + \frac{5}{12}\mathbf{k}_2 \right) \right) \\ \mathbf{y}_{i+1} &= \mathbf{y}_i + \Delta t \left(\frac{1}{4}\mathbf{k}_1 + \frac{3}{4}\mathbf{k}_2 \right) \end{aligned} \qquad \begin{array}{c|cc} 0 & \frac{1}{4} & -\frac{1}{4} \\ \frac{2}{3} & \frac{1}{4} & \frac{5}{12} \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array} \tag{4.93}$$

The implicit 2-stage Radau IA scheme is consistent and accurate of order $p = 3$, and thus satisfies requirement 1.

2-Stage Gauss (IRK 4):

The fourth-order Hammer-Hollingsworth scheme belongs to the class of methods that is based on the Gaussian quadrature formulas, which implies that $c_1..c_s$ are the zeros of the shifted Legendre polynomials of degree s and the order of accuracy is $2s$.

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}\left(t_{i+c_1},\, \mathbf{y}_i + \Delta t \left(a_{11}\mathbf{k}_1 + a_{12}\mathbf{k}_2 \right) \right) \\ \mathbf{k}_2 &= \mathbf{f}\left(t_{i+c_2},\, \mathbf{y}_i + \Delta t \left(a_{21}\mathbf{k}_1 + a_{22}\mathbf{k}_2 \right) \right) \\ \mathbf{y}_{i+1} &= \mathbf{y}_i + \Delta t \left(b_1\mathbf{k}_1 + b_2\mathbf{k}_2 \right) \end{aligned} \qquad \begin{array}{c|cc} \frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6} \\ \frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \tag{4.94}$$

The implicit 2-stage Gauss scheme is consistent and accurate of order $p = 4$, and thus satisfies requirement 1. Since this is an implicit algorithm with a nonsingular matrix $\mathbf{A}$, Equations (4.94) are next transformed into form (4.82), to reduce the influence of round off errors. From the two stages, the following system of nonlinear equations, with unknowns $\mathbf{z}$, is obtained.

$$F(\mathbf{z}) = \begin{Bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{Bmatrix} - \Delta t \begin{Bmatrix} a_{11}\mathbf{f}(\mathbf{z}_1) + a_{12}\mathbf{f}(\mathbf{z}_2) \\ a_{21}\mathbf{f}(\mathbf{z}_1) + a_{22}\mathbf{f}(\mathbf{z}_2) \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{0} \end{Bmatrix} \tag{4.95}$$

In order to solve such a system of nonlinear equations for the unknown stages $\mathbf{z}$, the well-known iterative Newton-Raphson algorithm is employed once again.

$$DF\left(\mathbf{z}^{(k)}\right) \Delta\mathbf{z}^{(k)} = -F\left(\mathbf{z}^{(k)}\right) \tag{4.96}$$

In Equation (4.96), the inverse of the mass matrix remains present. Therefore, lines 2 and 4 are left-multiplied by the mass matrix, which produces the following Jacobian.

$$DF\left(\mathbf{z}^{(k)}\right) = \begin{bmatrix} \mathbf{I} & -\Delta t\, a_{11}\mathbf{I} & \mathbf{0} & -\Delta t\, a_{12}\mathbf{I} \\ \Delta t\, a_{11}\mathbf{K}_t\left(\mathbf{z}_{11}^{(k)}\right) & \mathbf{M} + \Delta t\, a_{11}\mathbf{C} & \Delta t\, a_{12}\mathbf{K}_t\left(\mathbf{z}_{21}^{(k)}\right) & \Delta t\, a_{12}\mathbf{C} \\ \mathbf{0} & -\Delta t\, a_{21}\mathbf{I} & \mathbf{I} & -\Delta t\, a_{22}\mathbf{I} \\ \Delta t\, a_{21}\mathbf{K}_t\left(\mathbf{z}_{11}^{(k)}\right) & \Delta t\, a_{21}\mathbf{C} & \Delta t\, a_{22}\mathbf{K}_t\left(\mathbf{z}_{21}^{(k)}\right) & \mathbf{M} + \Delta t\, a_{22}\mathbf{C} \end{bmatrix} \tag{4.97}$$

With all the parts of Equation (4.96) available, the iterations can be initialized from $\mathbf{z}^{(k=1)} = \{\mathbf{0}\}$, which is a starting vector that is sufficiently close to the actual solution. Utilizing tensor products, the Newton-Raphson iterations can be summarized as follows.

$$\begin{aligned} &\left(\mathbf{I} - \Delta t\, \mathbf{A} \otimes \mathbf{f}_{,z}\left(\mathbf{z}^{(k)}\right)\right)\Delta\mathbf{z}^{(k)} = -\mathbf{z}^{(k)} + \Delta t\left(\mathbf{A} \otimes \mathbf{I}\right)\mathbf{f}\left(\mathbf{z}^{(k)}\right) \\ &\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} + \Delta\mathbf{z}^{(k)} \end{aligned} \tag{4.98}$$

As with any iterative method, the Jacobian matrix can be updated at every iteration or at the beginning of a time step, or kept constant at the initial value throughout the whole nonlinear dynamic analysis. Furthermore, it is possible to perform only a single Newton iteration, which is comparable to a linear implicit method. Once the iteration process on the linear system of Equation (4.98) has converged for time step t_{i+1}, the response quantities such as displacements, velocities, and accelerations are updated according to Equation (4.99). Finally, the repetition of this procedure for the total number of N time steps, with Newton-Raphson iterations in each such step, yields the sought solution. The implicit 2-stage Gauss method is summarized as pseudo-code in Fig. 4.32.

$$
\begin{aligned}
\Delta \mathbf{y} &= \sum_{n=1}^{s} \hat{b}_n \mathbf{z}_n \\
\mathbf{U}_{i+1} &= \mathbf{U}_i + \Delta \mathbf{y_1} \\
\dot{\mathbf{U}}_{i+1} &= \dot{\mathbf{U}}_i + \Delta \mathbf{y_2} \\
\ddot{\mathbf{U}}_{i+1} &= \ddot{\mathbf{U}}_i + \frac{\Delta \mathbf{y_2}}{\Delta t}
\end{aligned}
\tag{4.99}
$$

Initialize:

$\mathbf{U}_0 , \dot{\mathbf{U}}_0 , \ddot{\mathbf{U}}_0$

$\mathbf{y}_0 = \left\{ \mathbf{U}_0 , \dot{\mathbf{U}}_0 \right\}^T$

Analyze:

for $(i = 0,\ i < N,\ i{+}{+})$

$\mathbf{z}^{(k=1)} = \{\mathbf{0}\}$

do

$F^{(k)} = -\mathbf{z}$

$\mathbf{y}_{i+1} = \mathbf{y}_i + \mathbf{z}_1$ *(1st stage)*

$F^{(k)} += \Delta t \begin{Bmatrix} a_{11}\mathbf{f}\left(t_{i+c_1}, \mathbf{y}_{i+1}\right) \\ a_{21}\mathbf{f}\left(t_{i+c_1}, \mathbf{y}_{i+1}\right) \end{Bmatrix} \quad DF^{(k)}(1:4,1:2)$

$\mathbf{y}_{i+1} = \mathbf{y}_i + \mathbf{z}_2$ *(2nd stage)*

$F^{(k)} += \Delta t \begin{Bmatrix} a_{12}\mathbf{f}\left(t_{i+c_2}, \mathbf{y}_{i+1}\right) \\ a_{22}\mathbf{f}\left(t_{i+c_2}, \mathbf{y}_{i+1}\right) \end{Bmatrix} \quad DF^{(k)}(1:4,3:4)$

$\Delta \mathbf{z}^{(k)} = solve\left(DF^{(k)}\, \Delta \mathbf{z}^{(k)} = F^{(k)}\right)$

$\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} + \Delta \mathbf{z}^{(k)}$

while $(criterion > tol;\ k{+}{+})$

$\Delta \mathbf{y} = \hat{b}_1 \mathbf{z}_1 + \hat{b}_2 \mathbf{z}_2$

$\mathbf{U}_{i+1} = \mathbf{U}_i + \Delta \mathbf{y_1}$

$\dot{\mathbf{U}}_{i+1} = \dot{\mathbf{U}}_i + \Delta \mathbf{y_2}$

$\ddot{\mathbf{U}}_{i+1} = \ddot{\mathbf{U}}_i + \frac{\Delta \mathbf{y_2}}{\Delta t}$

end

Fig. 4.32 Implicit 2-stage Gauss (IRK4) method.

The continuity and uniformity of the displacement commands, which are generated during the iteration process, are again investigated by analyzing the two-degrees-of-freedom,

one-bay-frame model employing the different implicit Runge-Kutta integration methods. Fig. 4.30 shows close-ups of the displacement command signals at the controlled degree of freedom of the experimental element over three integration time steps for the different IRK methods.

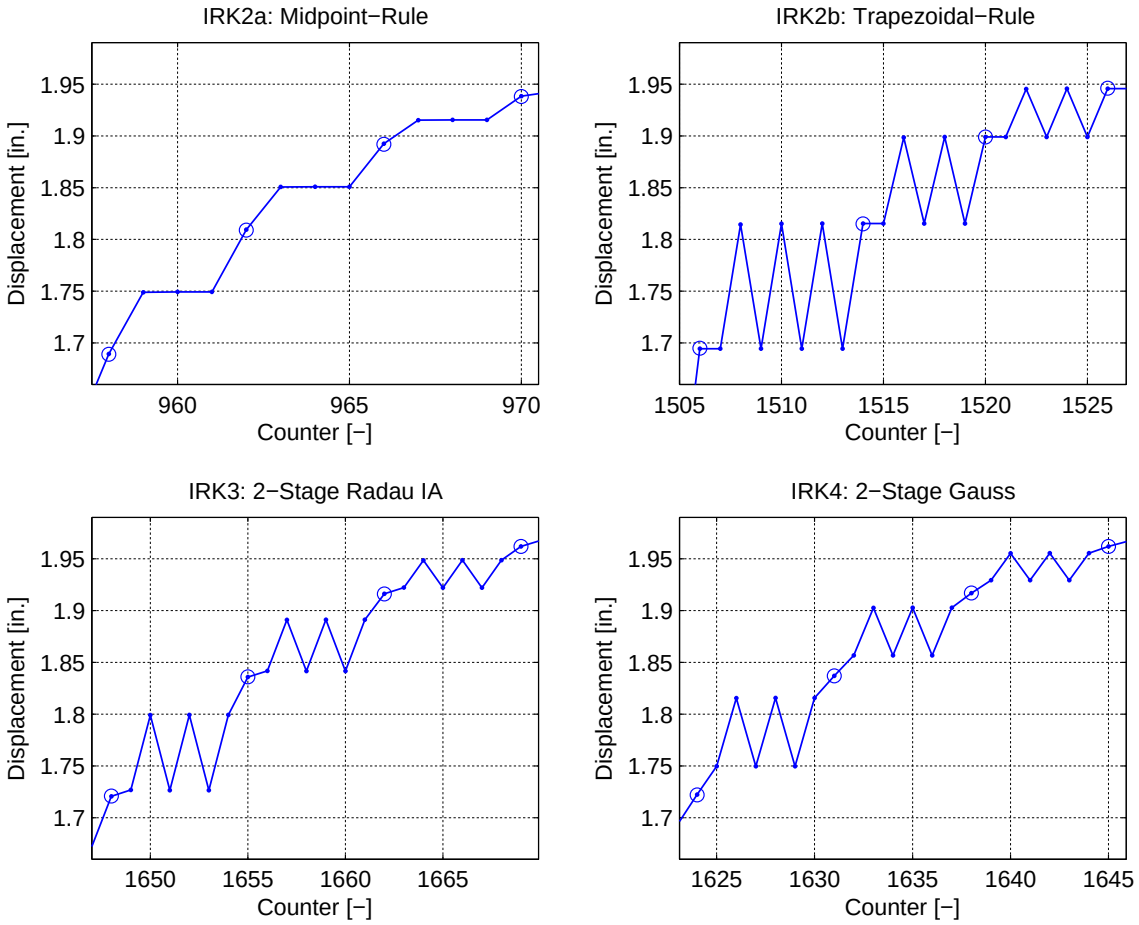


Fig. 4.33 IRK convergence comparison for one-bay-frame model.

As can be seen from the plots, none of the implicit Runge-Kutta methods produce strictly increasing and uniform displacement commands, violating both hybrid simulation requirements 6 and 7. In fact, all the implicit methods with two or more stages generate widely oscillating displacement increments during iteration. These oscillations occur because each stage is evaluated at a different time increment. So in every iteration, displacements are calculated at all the different t_{i+c_m}. This means that all the implicit Runge-Kutta methods with two or more stages cannot be used for hybrid simulation. In addition, due to the nonuniform displacement increments, the 1-stage midpoint-rule should be used for hybrid simulation only in combination with specialized event-driven algorithms, as it is the case for the INM1 direct integration method.

However, as with the higher-order ERK methods, the IRK methods are very compelling numerical integration schemes for the analysis of purely analytical structures because of their higher orders of accuracy.

4.5.3 Rosenbrock Methods (RRK)

In the previous section, it was shown that implicit Runge-Kutta methods cannot be used to perform the dynamic analysis in hybrid simulation because they need to solve the resulting nonlinear equations by Newton iteration. Thus, a different class of integration schemes, called Rosenbrock methods, is investigated next. Rosenbrock methods are a generalization of Linear-implicit Runge-Kutta methods that replace the nonlinear system of equations by a sequence of linear systems. They incorporate the Jacobian directly into the algorithm instead of evaluating it during the iteration process.

For nonautonomous problems, like the ones in structural dynamics, the general $s-stage$ Rosenbrock method is defined as follows (Hairer and Wanner 2002).

$$\begin{aligned}\mathbf{k}_m &= \mathbf{f}\left(t_{i+c_m},\, \mathbf{y}_i + \Delta t\sum_{n=1}^{m-1} a_{mn}\mathbf{k}_n\right) + \gamma_m \Delta t\, \mathbf{f}_{,t}\left(t_i,\, \mathbf{y}_i\right) + \Delta t\, \mathbf{f}_{,y}\left(t_i,\, \mathbf{y}_i\right)\sum_{n=1}^{m} \alpha_{mn}\mathbf{k}_n \quad \text{for m=1..s} \\ \mathbf{y}_{i+1} &= \mathbf{y}_i + \Delta t\sum_{n=1}^{s} b_n\mathbf{k}_n\end{aligned} \tag{4.100}$$

where the additional coefficients are given by

$$c_m = \sum_{n=1}^{m-1} a_{mn} \qquad and \qquad \gamma_m = \sum_{n=1}^{m} \alpha_{mn} \tag{4.101}$$

Similarly to Singly diagonally implicit Runge-Kutta (SDIRK) methods, Rosenbrock schemes with $\alpha_{mm} = \alpha \;\; \forall m$ are of special interest and can be expressed as follows.

$$\begin{aligned}\left(\mathbf{I} - \alpha\Delta t\, \mathbf{f}_{,y}\left(t_i,\, \mathbf{y}_i\right)\right)\mathbf{k}_m &= \mathbf{f}\left(t_{i+c_m},\, \mathbf{y}_i + \Delta t\sum_{n=1}^{m-1} a_{mn}\mathbf{k}_n\right) + \gamma_m \Delta t\, \mathbf{f}_{,t}\left(t_i,\, \mathbf{y}_i\right) \\ &\quad + \Delta t\, \mathbf{f}_{,y}\left(t_i,\, \mathbf{y}_i\right)\sum_{n=1}^{m-1} \alpha_{mn}\mathbf{k}_n \quad \text{for m=1..s} \\ \mathbf{y}_{i+1} &= \mathbf{y}_i + \Delta t\sum_{n=1}^{s} b_n\mathbf{k}_n\end{aligned} \tag{4.102}$$

As with the IRK methods, lines 2 and 4 are again left-multiplied by the mass matrix to eliminate the inversion of such a matrix and make the schemes applicable to problems with

singular mass matrices. In the above equations, the derivatives of the function $\mathbf{f}$ with respect to time t and to the solution vector $\mathbf{y}$ are approximated by the following expressions.

$$\mathbf{f}_{,t} = \frac{1}{\Delta t}\begin{Bmatrix} \mathbf{0} \\ \Delta\mathbf{P}_i - \Delta\mathbf{P}_{\mathbf{0},i} \end{Bmatrix} \qquad and \qquad \mathbf{f}_{,y} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{K_t}\left(\mathbf{y}_{\mathbf{1},i}\right) & \mathbf{C} \end{bmatrix} \tag{4.103}$$

The Rosenbrock methods are symbolized by the following modified Butcher array.

$$\begin{array}{c|c|c} \mathbf{c} & \mathbf{A} & \boldsymbol{\alpha} \\ \hline & \mathbf{b}^T & \end{array} \tag{4.104}$$

2-Stage Rosenbrock (RRK2):

$$\begin{aligned} \left(\mathbf{I} - \alpha\Delta t\,\mathbf{f}_{,y}\right)\mathbf{k}_1 &= \mathbf{f}\left(t_i,\, \mathbf{y}_i\right) + \alpha\Delta t\,\mathbf{f}_{,t} \\ \left(\mathbf{I} - \alpha\Delta t\,\mathbf{f}_{,y}\right)\mathbf{k}_2 &= \mathbf{f}\left(t_{i+c_2},\, \mathbf{y}_i + \Delta t\, a_{21}\mathbf{k}_1\right) \\ &\quad + \Delta t\,\mathbf{f}_{,y}\alpha_{21}\mathbf{k}_1 \\ \mathbf{y}_{i+1} &= \mathbf{y}_i + \Delta t\,\mathbf{k}_2 \end{aligned} \qquad \begin{array}{c|cc|cc} 0 & 0 & 0 & \alpha & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & -\alpha & \alpha \\ \hline & 0 & 1 & & \end{array} \tag{4.105}$$

Once the parameter $\alpha = 1 - \sqrt{2}/2$ is determined from the requirement that the Rosenbrock method should be L-stable, the remaining parameters are chosen such that all the order conditions for second-order accuracy are satisfied (Kaps & Wanner 1981). The L-stable, 2-stage Rosenbrock method is thus consistent and accurate of order $p = 2$, satisfying requirement 1.

3-Stage Rosenbrock (RRK3):

$$\begin{aligned} \left(\mathbf{I} - \alpha\Delta t\,\mathbf{f}_{,y}\right)\mathbf{k}_1 &= \mathbf{f}\left(t_i,\, \mathbf{y}_i\right) + \gamma_1\,\Delta t\,\mathbf{f}_{,t} \\ \left(\mathbf{I} - \alpha\Delta t\,\mathbf{f}_{,y}\right)\mathbf{k}_2 &= \mathbf{f}\left(t_{i+c_2},\, \mathbf{y}_i + \Delta t\, a_{21}\mathbf{k}_1\right) \\ &\quad + \gamma_2\,\Delta t\,\mathbf{f}_{,t} + \Delta t\,\mathbf{f}_{,y}\alpha_{21}\mathbf{k}_1 \\ \left(\mathbf{I} - \alpha\Delta t\,\mathbf{f}_{,y}\right)\mathbf{k}_3 &= \mathbf{f}\left(t_{i+c_3},\, \mathbf{y}_i + \Delta t\left(a_{21}\mathbf{k}_1 + a_{32}\mathbf{k}_2\right)\right) \\ &\quad + \gamma_3\,\Delta t\,\mathbf{f}_{,t} + \Delta t\,\mathbf{f}_{,y}\left(\alpha_{31}\mathbf{k}_1 + \alpha_{32}\mathbf{k}_2\right) \\ \mathbf{y}_{i+1} &= \mathbf{y}_i + \Delta t\left(\frac{1}{4}\mathbf{k}_1 + \frac{3}{4}\mathbf{k}_3\right) \end{aligned} \qquad \begin{array}{c|ccc|ccc} 0 & 0 & 0 & 0 & \alpha & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & \alpha_{21} & \alpha & 0 \\ \frac{2}{3} & \frac{2}{3} & 0 & 0 & \alpha_{31} & \alpha_{32} & \alpha \\ \hline & \frac{1}{4} & 0 & \frac{3}{4} & & & \end{array} \tag{4.106}$$

The parameter $\alpha = 0.4358665215$ is again determined from the requirement that the Rosenbrock method should be L-stable. While most of the other parameters can subsequently be determined from the order conditions for third-order accuracy, a_{31} and a_{32} remain free to choose

as long as their sum is equal to c_3. It was found that for $a_{31} = 2/3$ and $a_{32} = 0$ the L-stable, 3-stage Rosenbrock method produces the smoothest displacement commands. The remaining parameters then take the following values: $\alpha_{21} = 0.1817534184$, $\alpha_{31} = -0.3760889857$, and $\alpha_{32} = -0.2050663764$. The L-stable, 3-stage Rosenbrock method is consistent and accurate of order $p = 3$, satisfying requirement 1.

Initialize:

$\mathbf{U}_0$, $\dot{\mathbf{U}}_0$, $\ddot{\mathbf{U}}_0$

$\mathbf{y}_0 = \left\{\mathbf{U}_0,\ \dot{\mathbf{U}}_0\right\}^T$

Analyze:

for $(i = 0,\ i < N,\ i{+}{+})$

$$\mathbf{J} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{K_t}\left(\mathbf{y}_{1,i}\right) & \mathbf{C} \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{M} \end{bmatrix} - \alpha \Delta t\, \mathbf{J}$$

$\mathbf{y}_{i+1} = \mathbf{y}_i$ $\qquad \mathbf{b} = \mathbf{f}\left(t_i,\ \mathbf{y}_{i+1}\right) + \alpha \begin{Bmatrix} \mathbf{0} \\ \Delta \mathbf{P}_i \end{Bmatrix}$ $\qquad$ *(1st stage)*

$\mathbf{k}_1 = solve\left(\mathbf{A}\,\mathbf{k}_1 = \mathbf{b}\right)$

$\mathbf{y}_{i+1} = \mathbf{y}_i + \frac{1}{2}\Delta t\, \mathbf{k}_1$ $\qquad \mathbf{b} = \mathbf{f}\left(t_{i+1/2},\ \mathbf{y}_{i+1}\right) - \Delta t\, \mathbf{J}\, \alpha\, \mathbf{k}_1$ $\qquad$ *(2nd stage)*

$\mathbf{k}_2 = solve\left(\mathbf{A}\,\mathbf{k}_2 = \mathbf{b}\right)$

$\Delta \mathbf{y} = \Delta t\, \mathbf{k}_2$

$\mathbf{U}_{i+1} = \mathbf{U}_i + \Delta \mathbf{y_1}$

$\dot{\mathbf{U}}_{i+1} = \dot{\mathbf{U}}_i + \Delta \mathbf{y_2}$

$\ddot{\mathbf{U}}_{i+1} = \ddot{\mathbf{U}}_i + \frac{\Delta \mathbf{y_2}}{\Delta t}$

end

Fig. 4.34 2-stage Rosenbrock (RRK2) method.

The implementation of these two Rosenbrock methods requires, at each stage, the solution of a linear system of equations with matrix $\mathbf{I} - \alpha \Delta t\, \mathbf{f}_{,y}$, which is updated at the beginning of each time step. The second-order method is summarized as pseudo-code in Fig. 4.34. The implementation of the third-order method follows the same pattern.

The continuity and uniformity of the displacement commands, which are generated during the iteration process, are again investigated by analyzing the two-degrees-of-freedom, one-bay-frame model employing the different Rosenbrock integration methods. Fig. 4.35 shows close-ups of the displacement command signals at the controlled degree of freedom of the experimental element over three integration time steps for the different RRK methods.

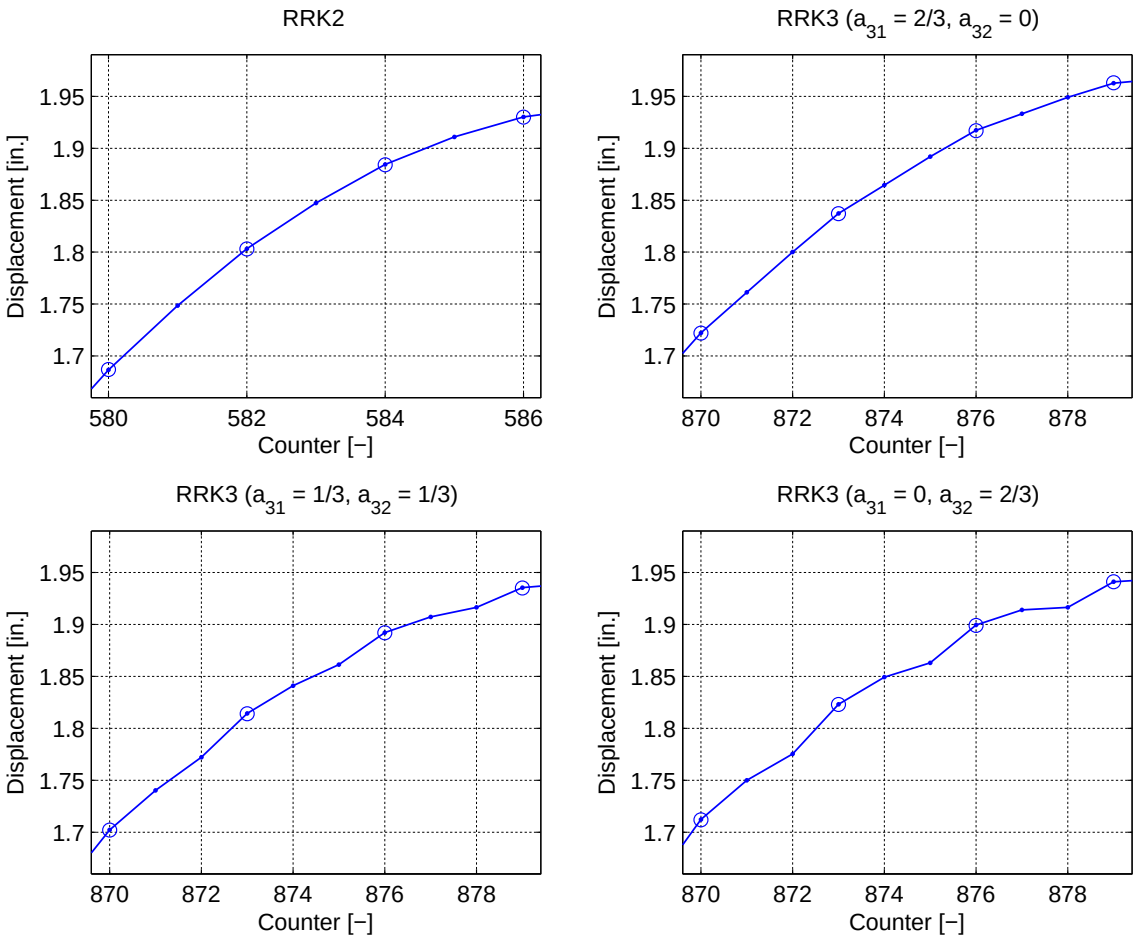


Fig. 4.35 RRK convergence comparison for one-bay-frame model.

The two plots on the top show the displacement commands, as received by the experimental column, for the two proposed Rosenbrock methods. For both the second- and the third-order schemes, the command signals are strictly increasing and very smooth. The two plots on the bottom of Fig. 4.35 show the displacement commands for two other choices of the free parameters a_{31} and a_{32} in the third-order method. Even though the command signals are still strictly increasing, the increments are clearly not as uniform as for the proposed set of

parameters. The two discussed Rosenbrock methods both satisfy requirements 6 and 7, and thus, are excellent integration schemes for hybrid simulation.

4.5.4 Stability

Since Runge-Kutta methods are one-step schemes they are essentially always stable. This means conditionally stable as long as a small enough time step is selected or unconditionally stable. However, the following concepts that are described next are utilized to investigate the extent of their stability.

A-Stability:

An initial value problem is A-stable if the region of absolute stability includes the entire left-half plane. This means that there are no stability restrictions for the test equation $\dot{y} = \lambda y$ $\forall\ \mathrm{Re}(\lambda) < 0,\ \Delta t > 0$. This is comparable to unconditional stability for direct integration methods. For Runge-Kutta methods, the stability function $R(z)$ (which determines the region of absolute stability, RAS: $|R(z)| < 1$) can be found from the following equations.

$$\begin{aligned} R(z) &= 1 + z\,\mathbf{b}^T (\mathbf{I} - z\,\mathbf{A})^{-1}\,\mathbf{e} \qquad (ERK,\ IRK) \\ R(z) &= 1 + z\,\mathbf{b}^T (\mathbf{I} - z\,\mathbf{B})^{-1}\,\mathbf{e} \qquad (RRK) \end{aligned} \tag{4.107}$$

where $z = \Delta t\,\lambda$, $\mathbf{e}$ is a column vector of all ones and $\mathbf{B} = [a_{mn} + \alpha_{mn}]$.

L-Stability:

A method is L-stable if it fulfills the following conditions: (1) it must be A-stable and (2) the limit of the stability function must approach zero as z approaches infinity.

$$\lim_{z \to \infty} R(z) = 0 \tag{4.108}$$

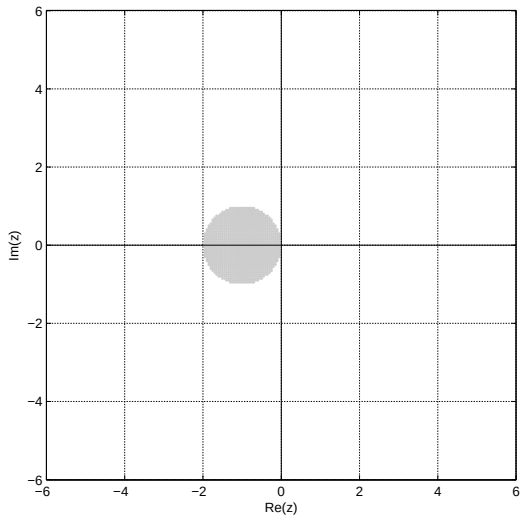
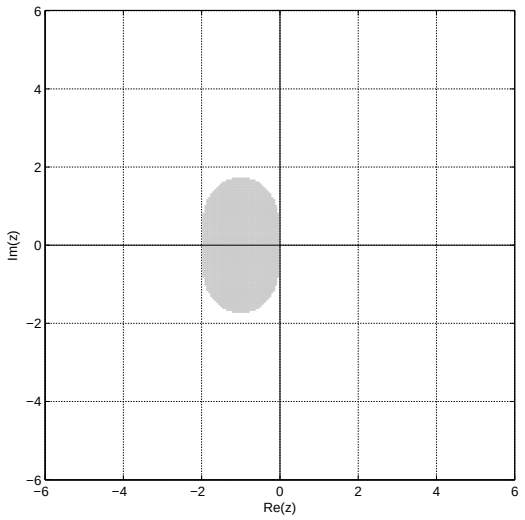
L-stable methods have the important property that they damp out the transient phases of the stiff components very rapidly. This stability requirement is thus comparable with the requirement that an integration scheme should provide some algorithmic damping, which was postulated for the direct integration methods.

B-Stability:

Finally, for implicit Runge-Kutta methods the stability of nonlinear differential equations is determined by the B-Stability property. Runge-Kutta methods are B-stable if they are algebraically stable. For algebraic stability, the following two matrices have to be positive semi-definite.

$$B = diag(b)$$
$$M = BA + A^T B - b\,b^T \tag{4.109}$$

Regions of Absolute Stability (RAS):

<table>
<tr>
<td>Euler (ERK1):
$R(z) = 1 + z$
$\rightarrow$ not A - stable</td>
<td>Runge (ERK2):
$R(z) = 1 + z + \frac{1}{2}z^2$
$\rightarrow$ not A - stable</td>
</tr>
<tr>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>Heun (ERK3):
$R(z) = 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3$
$\rightarrow$ not A - stable</td>
<td>Classic RK and 3/8 rule (ERK4):
$R(z) = 1 + z + \frac{1}{2}z^2 + \frac{1}{6}z^3 + \frac{1}{24}z^4$
$\rightarrow$ not A - stable</td>
</tr>
</table>

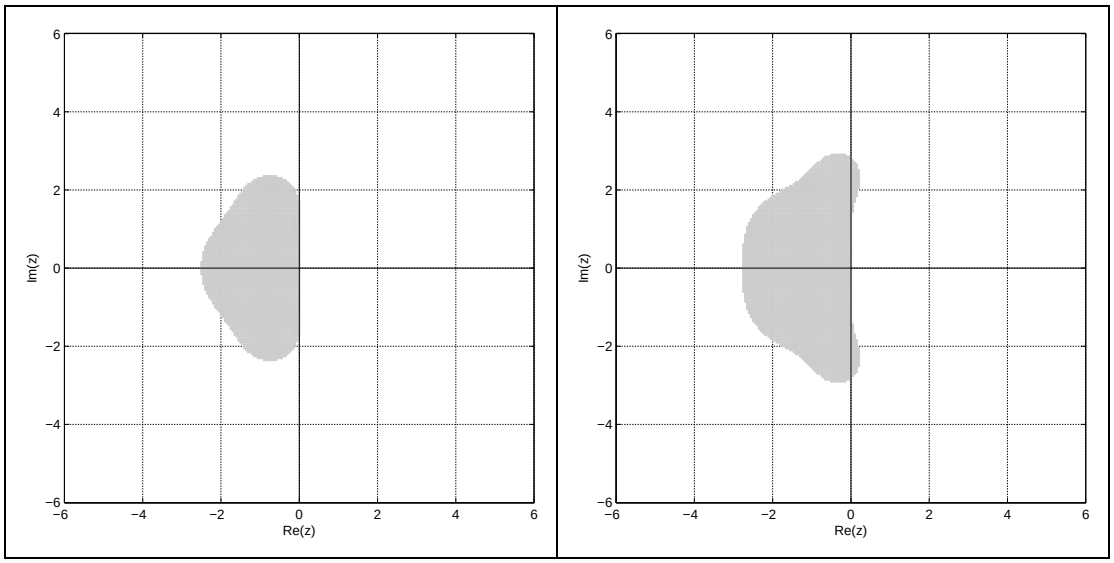


Fig. 4.36 Regions of Absolute Stability (RAS) for ERK methods.

Midpoint-Rule (IRK2a):

$$R(z)=\frac{2+z}{2-z}$$

$$R(\infty)=-1$$

$$\mathbf{B}=1,\ \mathbf{M}=0$$

$\rightarrow$ *A - stable and B - stable*

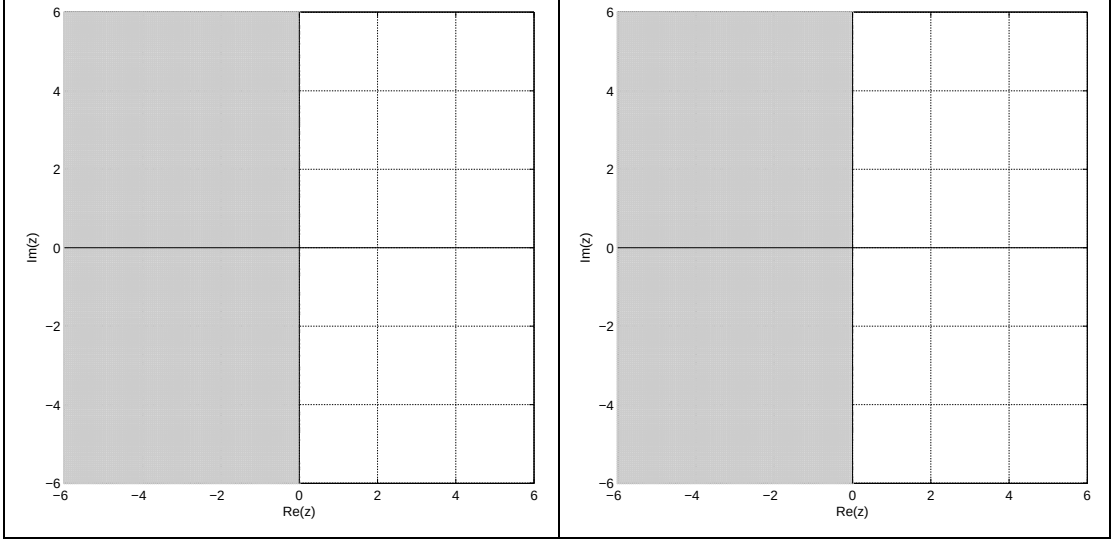
Trapezoidal-Rule (IRK2b):

$$R(z)=\frac{2+z}{2-z}$$

$$R(\infty)=-1$$

$$\mathbf{B}=\begin{bmatrix}\frac{1}{2} & 0\\ 0 & \frac{1}{2}\end{bmatrix},\ \mathbf{M}=\begin{bmatrix}-\frac{1}{4} & 0\\ 0 & \frac{1}{4}\end{bmatrix}$$

$\rightarrow$ *A - stable*



2-Stage Radau IA (IRK3):

$$R(z)=\frac{6+2z}{6-4z+z^2}$$

$$R(\infty)=0$$

$$\mathbf{B}=\begin{bmatrix}\frac{1}{4} & 0\\ 0 & \frac{3}{4}\end{bmatrix},\ \mathbf{M}=\begin{bmatrix}\frac{1}{16} & -\frac{1}{16}\\ -\frac{1}{16} & \frac{1}{16}\end{bmatrix}$$

$\rightarrow$ *A - stable, L - stable and B - stable*

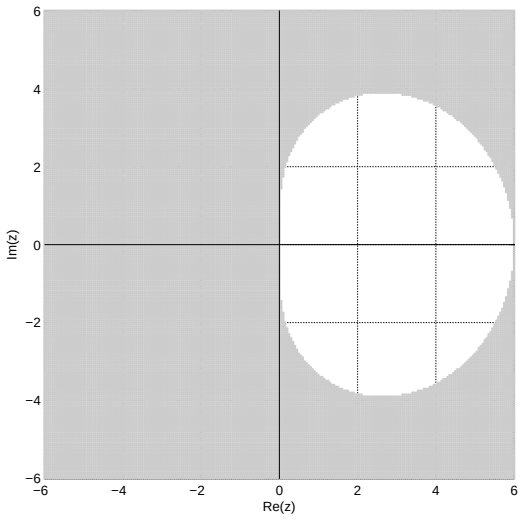
2-Stage Gauss (IRK4):

$$R(z)=\frac{12+6z+z^2}{12-6z+z^2}$$

$$R(\infty)=1$$

$$\mathbf{B}=\begin{bmatrix}\frac{1}{2} & 0\\ 0 & \frac{1}{2}\end{bmatrix},\ \mathbf{M}=\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}$$

$\rightarrow$ *A - stable and B - stable*



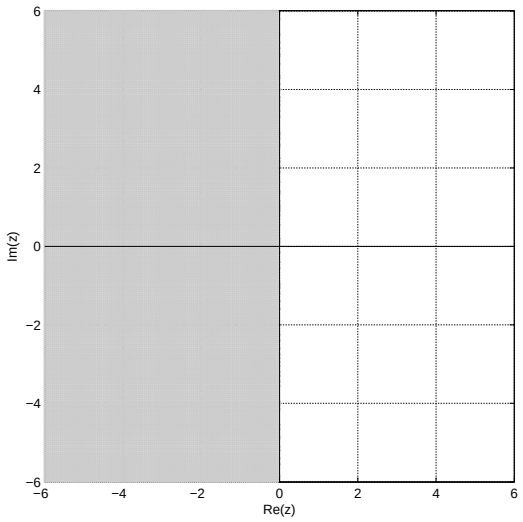


Fig. 4.37 Regions of Absolute Stability (RAS) for IRK methods.

2-Stage Rosenbrock (RRK2):

$$R(z)=\frac{4-4z+4\sqrt{2}z}{4-8z+4\sqrt{2}z+6z^2-4\sqrt{2}z^2}$$

$$R(\infty)=0$$

$$\mathbf{B}=\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}$$

$$\mathbf{M}=\begin{bmatrix}0 & 0.2071\\ 0.2071 & -0.4142\end{bmatrix}$$

$\rightarrow$ *A - stable and L - stable*

3-Stage Rosenbrock (RRK3):

$$R(z)=\frac{-12.077+3.715z+2.870z^2}{-12.077+15.791z-6.883z^2+z^3}$$

$$R(\infty)=0$$

$$\mathbf{B}=\begin{bmatrix}0.25 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0.75\end{bmatrix}$$

$$\mathbf{M}=\begin{bmatrix}0.1554 & 0 & 0.0304\\ 0 & 0 & -0.1538\\ 0.0304 & -0.1538 & 0.0913\end{bmatrix}$$

$\rightarrow$ *A - stable and L - stable*

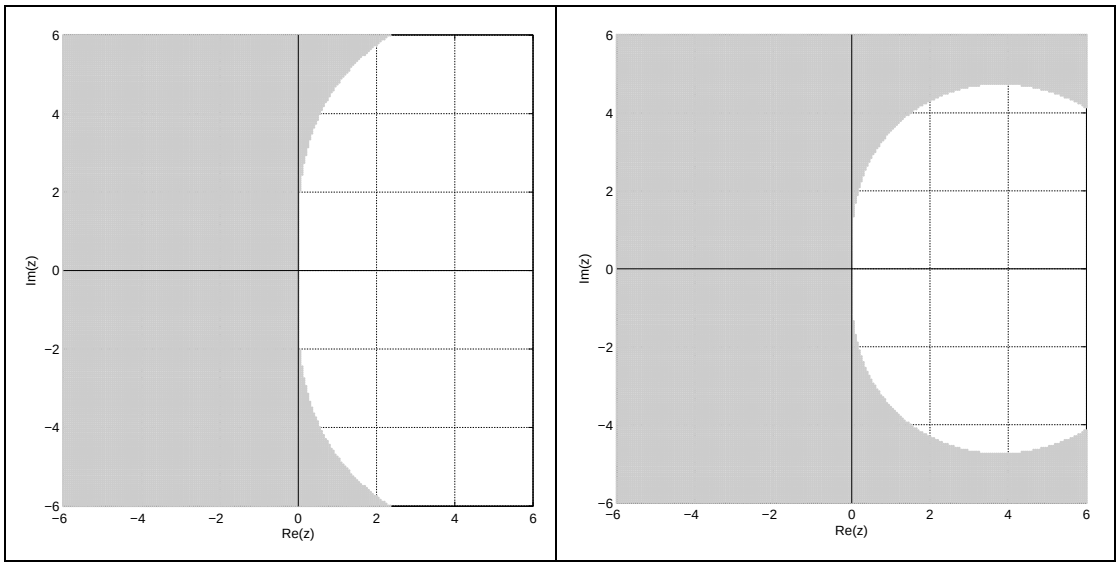


Fig. 4.38 Regions of Absolute Stability (RAS) for RRK methods.

4.5.5 Accuracy: Numerical Dissipation and Dispersion

As with the direct integration methods, the accuracy of the Runge-Kutta integration schemes is again evaluated in terms of the two measures, numerical dissipation and dispersion. Once again these accuracy measures are obtained from simulation results. Thus, free vibration tests of a linear-elastic single-degree-of-freedom system are performed for all the Runge-Kutta methods, in order to determine their numerical dissipation and dispersion properties. The SDOF system has a period of $T = 1\text{sec}$ and is excited by an initial displacement of $\mathbf{U_0} = 1$ in. and an initial velocity of $\dot{\mathbf{U}}_\mathbf{0} = 0$ in./sec, meaning that $\mathbf{y}_0 = \{1,\ 0\}^T$ becomes the starting vector for all the Runge-Kutta methods.

The next figure presents comparisons of the two accuracy measures, that is, the algorithmic damping ratios and the relative period errors, for the relevant Runge-Kutta integration methods. It can be seen that the third-order ERK3 method provides a large amount of algorithmic energy dissipation in the mid-frequency modes and shortens the modes over the whole frequency range. The fourth-order ERK4 method provides no numerical damping in the low-frequency modes but the amount of algorithmic damping increases rapidly for the high-frequency modes. In addition, the scheme is very accurate within the stability limits, meaning that the relative period errors are small. The ERK4 method elongates the low-frequency modes and shortens the high-frequency modes. For the implicit Runge-Kutta methods, the second-order ERK2a method achieves the same accuracy as the second-order implicit Newmark INM method.

This means that the scheme provides no numerical energy dissipation and elongates the modes over the whole frequency range. It can be seen that this algorithm is less accurate than the higher-order methods, since it elongates periods significantly more. The third-order IRK3 integrator achieves quite low relative period errors while providing the desired algorithmic energy dissipation in the high-frequency modes because of the algorithms L-stability. It is obvious that the fourth-order IRK4 method is the most accurate among the discussed schemes but unfortunately does not provide any numerical energy dissipation to damp out high-frequency oscillations. Finally, the second- and third-order Rosenbrock methods achieve good accuracy with relatively small period errors and the desired algorithmic damping in the high-frequency modes due to their L-stability.

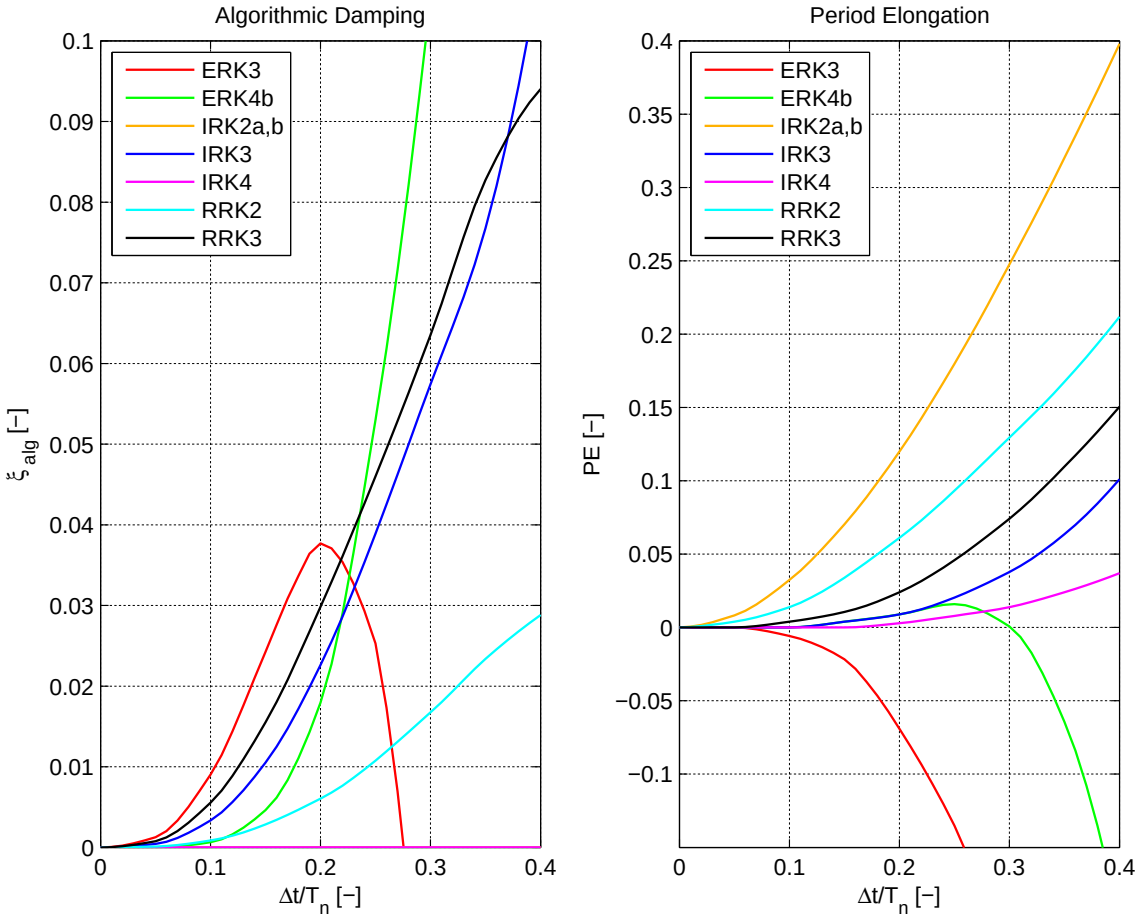


Fig. 4.39 Algorithmic damping ratios and relative period errors for Runge-Kutta methods.

All of the Runge-Kutta methods that have been presented above and that are summarized in the next subsection have so far only been implemented utilizing Matlab (Mathworks). However, there are plans to implement these Runge-Kutta methods into the OpenSees finite element software in the near future.

4.5.6 Summary of Special Requirements

Table 4.4 Summary of special requirements for Runge-Kutta integration methods.

Requirement	*ERK3*	*ERK4b*	*IRK2a*	*IRK2b*	*IRK3*	*IRK4*	*RRK2*	*RRK3*
1. $p \geq 2$ accurate	$p = 3$	$p = 4$	$p = 2$	$p = 2$	$p = 3$	$p = 4$	$p = 2$	$p = 3$
2. few function calls per Δt	3	4					2	3
3. unconditionally stable			✓	✓	✓	✓	✓	✓
4. constant $\mathbf{k}_{\mathbf{el},\exp}$	na	na	✓	✓	✓	✓	✓	✓
5. algorithmic damping	✓	✓			✓		✓	✓
6. strictly incr./decreasing $\Delta\mathbf{U}$	✓		✓				✓	✓
7. uniform $\Delta\mathbf{U}$	✓						✓	✓
8. real-time compatible	✓						✓	✓

4.6 SUMMARY

A wide range of integration methods and their applicability to hybrid simulation were presented in this chapter. Both the class of direct integration methods and the class of Runge-Kutta methods were investigated. After a brief introduction to the spatial and time discretization of the structural dynamics problem, special integration scheme properties required for the reliable, accurate and fast execution of hybrid simulations were introduced. It has been shown that explicit, noniterative methods are well suited for hybrid simulation and easy to implement, as long as it is possible to satisfy their stability limits. On the other hand, it has been demonstrated that unconditionally stable implicit methods should be used only for hybrid simulation under one of two conditions. Either the number of iterations in the equilibrium solution algorithms can be fixed at a constant value, or the event-driven algorithms described in the next chapter need to adjust the time interval over which the displacement increments are imposed during simulation. In addition, two special types of integration schemes, namely the operator-splitting and the Rosenbrock methods, were discussed. These types of schemes are capable of providing unconditional stability without the need for iterative equilibrium solution processes. For the operator-splitting schemes the generalized-alpha-OS and the modified generalized-alpha-OS methods were newly introduced. It has also been shown that for hybrid simulation the integration schemes should preferably provide some amount of algorithmic energy dissipation in the high-frequency modes to damp out any spurious oscillations that might get excited by experimental errors. Finally, the integration methods have been summarized in form of pseudo-code to assist with the implementation into software.

5 Event-Driven Strategies in Hybrid Simulation

5.1 INTRODUCTION

In the finite element analysis approach to hybrid simulation there are three key components required to perform experimental tests. The previous chapter focused on the first key component, the finite element analysis software itself. In particular, specialized time-stepping algorithms and their application to hybrid simulation were investigated. The second key component is the middleware that allows any finite element analysis software to be connected to a laboratory, and which was discussed in detail in Chapter 3. Consequently, this chapter is concerned with investigating methods and strategies for the third key component, meaning the control and data-acquisition systems in a laboratory.

Today's state of the art servo-hydraulic control systems, which are part of the third key component, run at sampling rates of the order of 1 kHz and above. This means that for the signal generation of the actuator commands, new displacement values are required at very short, deterministic time intervals. On the other hand, the numerical integration of the equations of motion, which is performed in the finite element analysis software, can be very time consuming. In addition, the computation times required to advance the numerical solutions to the next step are generally nondeterministic because they depend on events such as yielding, buckling, fracture, or collapse of structural elements. Hence, the computational driver and the servo-hydraulic control system run at different time rates, where the former is nondeterministic and the latter is deterministic. However, it is possible to synchronize these two processes by placing a predictor-corrector algorithm between the computational driver and the servo-hydraulic control system. It is important to notice that this synchronization predictor-corrector algorithm is implemented on a real-time digital signal processor in the laboratory (as part of the third key component) and is therefore not to be confused with integration predictor-corrector methods that are utilized to solve the equations of motion.

The predictor-corrector algorithms that are utilized for synchronization operate as follows. While the finite element analysis software is solving the equations of motion for the trial

displacements at the next integration time step, the predictor-corrector algorithm generates displacement commands for the servo-hydraulic control system (at such system's deterministic time intervals) by forward prediction of the displacement paths. Once the trial displacements at the next integration time step or substep are received from the finite element analysis software, the predictor-corrector algorithm switches to generating displacement commands (still at the control system's deterministic sampling rate) correcting towards these new targets. As long as the computation time, which is required to generate these predicted and corrected displacement commands, is shorter than the given sampling time of the servo-hydraulic control system, the actuators can be moved continuously, guaranteeing test continuity.

With the recent development of geographically distributed hybrid simulations, the difference in the time rates of the integration and actuator control processes becomes even greater due to the additional randomness (nondeterminism) of the network transmission times. Thus, previously developed extrapolation-interpolation methods (Nakashima and Masaoka 1999; Mosqueda 2003) are no longer ideal. The first problem is that the accuracy of such techniques rapidly deteriorates the further out the displacements are predicted. The second drawback is a large discrepancy between displacement commands while switching from extrapolation to interpolation. This sudden displacement jump prevents the actuators from moving smoothly and creates unrealistic velocities in the switching interval. Moreover, extrapolation and interpolation procedures are not based on physical laws but rather on pure mathematics.

To improve the performance of the synchronization predictor-corrector algorithms and to reduce or eliminate the aforementioned problems, a range of new predictor-corrector algorithms are proposed and discussed in this chapter. In particular, the newly developed algorithms should be more accurate than the existing ones, while concurrently reducing or eliminating the switching interval inconsistencies. Since some of the proposed predictor-corrector algorithms make use of past velocities and past accelerations, in addition to displacements, they depend on the integration methods (see Chapter 4), which produce such response quantities. Furthermore, several adaptive, event-driven strategies are investigated. These event-driven methods do not run synchronously, where operations are coordinated under the centralized control of a fixed-rate clock signal, but are executing asynchronously, meaning that no global clock exists and simulation time-steps or integration time-steps are adjusted during simulation according to some suitable performance criteria.

The next section focuses on the review of previous developments. Specifically, the work on extrapolation-interpolation methods facilitating continuous testing and the development of the three-loop architecture for distributed testing are emphasized. Afterwards, the theory on existing and newly proposed predictor-corrector algorithms is presented, including a brief summary of advantages, disadvantages, and recommendations for each algorithm. The adaptive, asynchronous, event-driven strategies that adjust parameters, such as time steps and actuator velocities, over the course of a hybrid simulation are investigated last.

5.2 PREVIOUS DEVELOPMENTS

5.2.1 Discontinuous vs. Continuous Hybrid Simulation

Conventional hybrid simulation testing techniques load the test specimens using a hold and ramp procedure. This means that a transfer system imposes the trial displacement commands (calculated by the computational driver) on the physical test specimen in a ramp-wise manner. Once the transfer system reaches these target displacements, they are held constant until the work conjugates (resisting forces) have been measured and the next trial displacement commands have been computed (see Fig. 5.1).

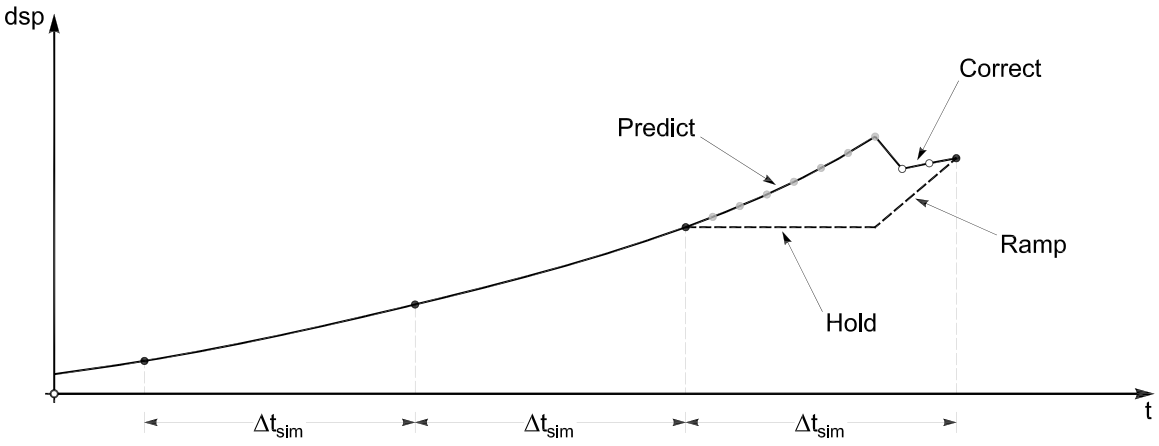


Fig. 5.1 Hold and ramp procedure vs. continuous testing procedure.

Although this procedure permits close observation of the response behavior of a structure, it has two major drawbacks. The first problem is that during the hold phase the test specimen undergoes force relaxation, which will lead to inaccurate resisting force measurements. The second disadvantage is that many new types of structural components, such as base-isolation

devices and dampers, have velocity-dependent behavior that a ramp and hold testing procedure cannot correctly capture. To deal with these problems two alternative hybrid simulation testing-methods have been developed. In the method known as continuous testing, the specimen is loaded at a slow but continuous rate, which solves the problem of force relaxation. Such continuous testing was first achieved by using an implicit integration algorithm and a direct force feedback using analog circuits (Thewalt and Mahin 1987). The second method is real-time testing, which loads the test specimen continuously at the appropriate velocities (accounting for similitude effects) and therefore captures rate-dependent material behaviors (Nakashima et al. 1992; Horiuchi et al. 1996; Nakashima and Masaoka 1999; Magonette 2001; Shing et al. 2002).

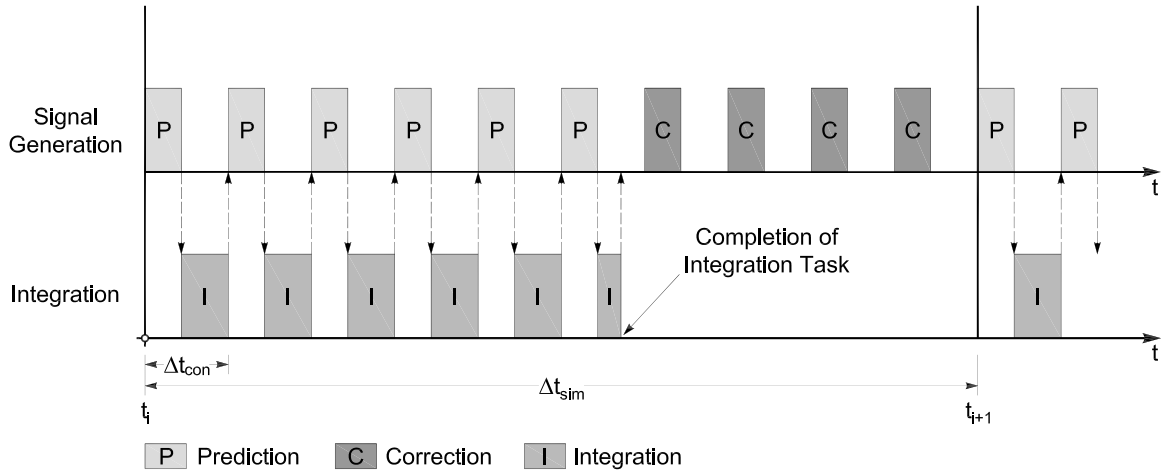


Fig. 5.2 Execution sequence in the case of one single digital signal processor.

To be able to run tests continuously, or in real time, most of these hybrid simulation methods use multi-rate approaches. This implies that both the integration of the equations of motion and the generation of displacement command signals are separated because they run at different time rates. In Nakashima and Masaoka's (1999) algorithm, both tasks run on a digital signal processor (DSP), which is a real-time processor. Therefore, in order to send uninterrupted command signals to the digital control system, the signal generation task (which has higher priority), has to execute every millisecond. However, since the signal generation takes less than one millisecond, the remaining time is used to solve the equations of motion. This means that while the integrator is calculating the trial displacements for the next time step, the DSP is switching forth and back between the two tasks; then, once the new trial displacements are available, only the signal generation tasks are executed (see Fig. 5.2). The proposed algorithm

applies polynomial extrapolation and interpolation using past displacement values to produce the command signals for the transfer system.

5.2.2 Three-Loop Hardware Architecture

The latest development in hybrid simulation is the geographical distribution of experimental and analytical portions of a structure within a network of laboratories and computational sites, a method first proposed by Campbell and Stojadinovic (1998). Mosqueda (2003) found that by separating the integration of the equations of motion and the signal generation for the transfer system onto two machines instead of one, both real-time testing and continuous geographically distributed testing is made possible. Furthermore, as can be seen from Fig. 5.3, it is no longer necessary to interrupt the response analysis task in order to execute the signal generation task, because they run on two different machines. Consequently, since the two processes are executed on two separate computers, fast data transfer between those machines is crucial. This is made possible by using a shared memory network, which mirrors data almost instantaneously among computers. This new three-loop hardware architecture is therefore able to dedicate more computation time towards the integration task, which makes it possible to increase the complexity of the analytical portions of the hybrid model.

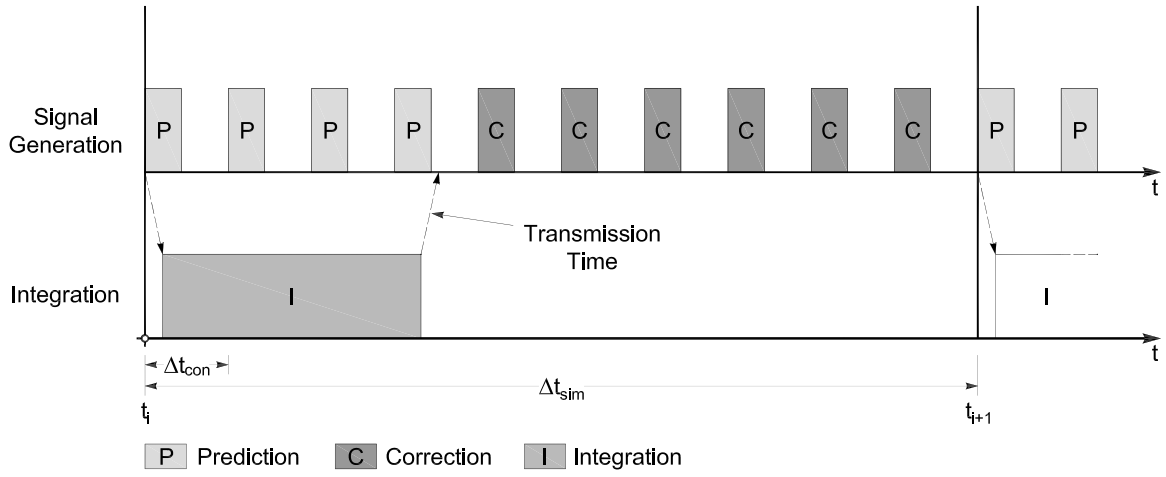


Fig. 5.3 Execution sequence in the case of two machines.

The three-loops in the new hardware architecture proposed by Mosqueda (2003) host the following tasks (see Fig. 5.4):

1. The outermost loop hosts the computational driver with the integration method as well as the OpenFresco middleware framework. In a real-time test, this task is executed at the

integration time-step interval, Δt_{int}, of the computational driver (0.005–0.02 sec). However, for geographically distributed testing, such a task has to be slowed down, allowing the simulation time-step interval, Δt_{sim}, to be controlled by the network speed (up to 3 sec) rather than by the time needed for the calculations.

2. The innermost loop of the architecture hosts the control and data-acquisition systems with their software. Any controller, such as a PID- (Proportional-Integral-Derivative), PIDF- (Proportional-Integral-Derivative-Feedforward), SM- (Sliding-Mode) controller etc., can be employed to command the actuators of the transfer system. The update rate of many such controllers is on the order of 1 kHz or higher, meaning that new command signals need to be sent to the actuators at the deterministic controller time-step interval, Δt_{con}.
3. The intermediate loop hosts the predictor-corrector algorithm that connects the two other processes running at different time rates. Similar to Nakashima and Masaoka's (1999) algorithm, it generates sub-step commands for the faster running control system using the trial response quantities from the slower running integration operator. The difference lies in the implementation of this predictor-corrector algorithm. Since in a geographically distributed test random network delays are introduced, Mosqueda (2003) utilizes an event-driven algorithm (a finite-state machine) to deal with these uncertainties.

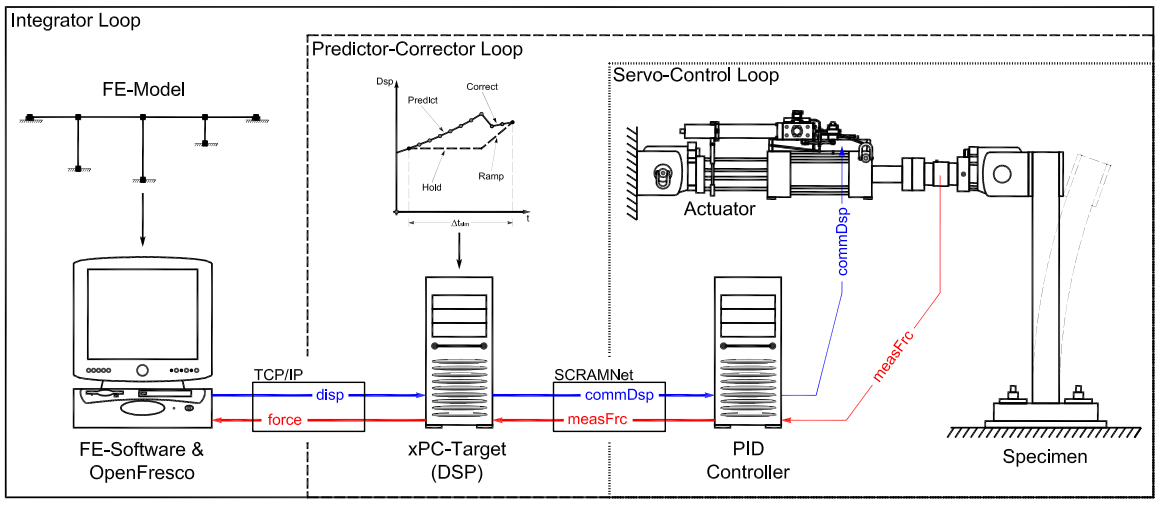


Fig. 5.4 Three-loop hardware architecture.

The advantage of this complete separation of tasks is its modularity, flexibility, and extensibility; thus, different computational drivers can be utilized and developed without concern for the synchronization with the control and data-acquisition systems. Similarly, control

algorithms can be developed independent of the integration methods. In addition, different predictor-corrector algorithms for the synchronization task can be developed. Finally, the division also facilitates the distribution of the three processes on three different machines, which allows the computational driver, meaning the finite element analysis software, to be placed anywhere on the network. This abstraction and encapsulation of the different tasks is similar to the object-oriented design methodologies that were used to develop the OpenFresco software framework (see Chapter 3).

5.3 THEORY ON PREDICTOR-CORRECTOR ALGORITHMS

5.3.1 Existing Algorithms

The predictor-corrector algorithms, which are used by other researchers such as Nakashima et al. (1999) and Mosqueda (2003), are based on extrapolation and interpolation, using displacement values at past integration points. The polynomials utilized are Lagrange polynomials of first-, second- or third-order (Burden and Faires 2001). Nakashima found that the third-order extrapolation/interpolation achieves the best accuracy given that computation time should remain short.

Prediction (Pn):

The extrapolated displacements are expressed in terms of a Lagrange polynomial using the last n displacement values.

$$dsp_p(x) = \sum_{k=-n}^{0} f_k \, P_{n,k}(x) \tag{5.1}$$

where f_k are the displacements previously calculated by the integrator and $x \in [0, 0.8]$. The prediction is limited to a maximum of 80% of the simulation time step in order to leave a minimum of 20% of the simulation time step for the correction towards the next trial displacement. The Lagrange functions $P_{n,k}$ up to a third-order polynomial are given below.

Table 5.1 Predictor Lagrange functions up to order 3.

$P_{1,0} = x+1$	$P_{2,0} = \frac{1}{2}(x+1)(x+2)$	$P_{3,0} = \frac{1}{6}(x+1)(x+2)(x+3)$
$P_{1,-1} = -x$	$P_{2,-1} = -x(x+2)$	$P_{3,-1} = -\frac{1}{2}x(x+2)(x+3)$
-	$P_{2,-2} = \frac{1}{2}x(x+1)$	$P_{3,-2} = \frac{1}{2}x(x+1)(x+3)$
-	-	$P_{3,-3} = -\frac{1}{6}x(x+1)(x+2)$

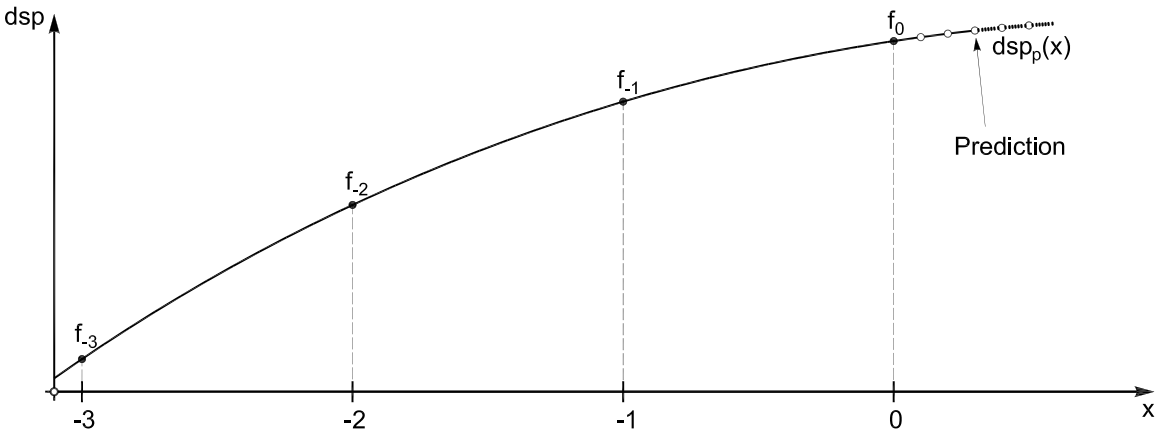


Fig. 5.5 Predictor using a third-order Lagrange polynomial.

Similar to Nakashima and Masaoka (1999), the accuracy of the predictors is evaluated by assuming that the earthquake response of a structure is a sinusoidal-like vibration. Using this approximation, the displacements are expressed in terms of amplitude A and circular frequency ω as follows. It is important to note that the circular frequency of the sinusoidal vibration ω is provided in the integration time scale and not the simulation time scale.

$$f(t_{int}) = A \sin(\omega t_{int}) \tag{5.2}$$

The discrete displacements at the previous simulation time steps, as needed in the predictor formula (5.1), are therefore given by the following expression.

$$f_k = A \sin(\omega t_{int} - \omega \Delta t_{int}(x-k)) \tag{5.3}$$

After the substitution of Equation (5.3) into (5.1) and the trigonometric expansion of the resulting formula, the following result is obtained.

$$\begin{aligned} dsp_p(x) &= \sum_{k=-n}^{0} A \sin\left(\omega t_{int} - \omega \Delta t_{int}(x-k)\right) P_{n,k}(x) \\ &= A R_d \sin\left(\omega t_{int} + \phi\right) \end{aligned} \tag{5.4}$$

where:

$$\begin{aligned} R_d &= \sqrt{C_S^2 + C_C^2}\ , \quad \phi = \arctan\left(\frac{C_C}{C_S}\right) \\ C_S &= \sum_{k=-n}^{0} \cos\left(\omega \Delta t_{int}(x-k)\right) P_{n,k}(x) \\ C_C &= -\sum_{k=-n}^{0} \sin\left(\omega \Delta t_{int}(x-k)\right) P_{n,k}(x) \end{aligned} \tag{5.5}$$

As can be seen, by comparing formula (5.2) with formula (5.4) the predicted displacements diverge from the exact displacements. The difference can be expressed in terms of an amplitude change R_d and a phase shift ϕ. The amplitude change and phase shift are therefore used as a measure of accuracy. In general, the accuracy of any predictor will decrease the further out a predicted value is sought and the larger the integration time interval is with respect to the excitation period. Obviously, the integration time interval needs to be small enough to guarantee stability and accuracy of the direct integration method used.

Correction (Cn):

The interpolated displacements are expressed in terms of a Lagrange polynomial similar to the one used for extrapolation:

$$dsp_c(x) = \sum_{k=-n+1}^{1} f_k C_{n,k}(x) \tag{5.6}$$

where f_k are the displacements previously calculated by the integrator and $x \in [0, 1]$. The Lagrange functions $C_{n,k}$ up to a third-order polynomial are given in Table 5.2. Again, Equation (5.3) is substituted into (5.6) and the result is trigonometrically expanded to obtain the following corrector formula.

$$dsp_c(x) = \sum_{k=-n+1}^{1} A \sin\left(\omega t_{int} - \omega \Delta t_{int}(x-k)\right) C_{n,k}(x) = A R_d \sin\left(\omega t_{int} + \phi\right) \tag{5.7}$$

where:

$$R_d = \sqrt{C_S^2 + C_C^2}\ , \quad \phi = \arctan\left(\frac{C_C}{C_S}\right)$$

$$C_S = \sum_{k=-n+1}^{1} \cos\left(\omega \Delta t_{int}(x-k)\right) C_{n,k}(x) \tag{5.8}$$

$$C_C = -\sum_{k=-n+1}^{1} \sin\left(\omega \Delta t_{int}(x-k)\right) C_{n,k}(x)$$

Table 5.2 Corrector Lagrange functions up to order 3.

$C_{1,1} = x$	$C_{2,1} = \frac{1}{2}x(x+1)$	$C_{3,1} = \frac{1}{6}x(x+1)(x+2)$
$C_{1,0} = -(x-1)$	$C_{2,0} = -(x-1)(x+1)$	$C_{3,0} = -\frac{1}{2}(x-1)(x+1)(x+2)$
-	$C_{2,-1} = \frac{1}{2}(x-1)x$	$C_{3,-1} = \frac{1}{2}(x-1)x(x+2)$
-	-	$C_{3,-2} = -\frac{1}{6}(x-1)x(x+1)$

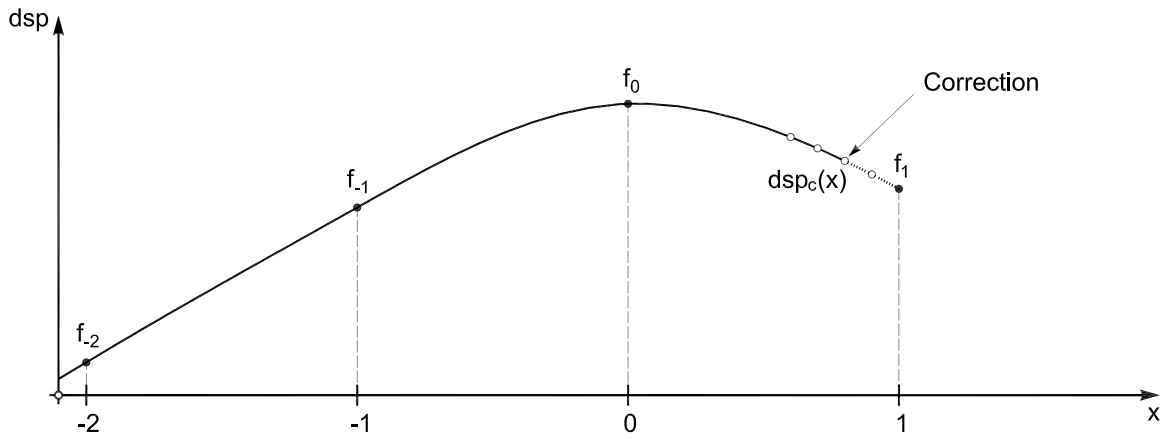


Fig. 5.6 Corrector using a third-order Lagrange polynomial.

As for the predictor formulas, the accuracy measures used are the change in amplitude and the phase shift. Since the corrector formulas make use of the trial displacements at the new time step or substep, they are overall more accurate than the predictor formulas. Furthermore, while they return the exact values at $t_{int,i}$ and $t_{int,i} + \Delta t_{int}$, they are the least accurate around the middle of a time step. Finally, similar to the predictor formulas, the accuracy decreases as the integration time interval increases with respect to the excitation period.

5.3.2 Algorithms Using Last Predicted Displacement

Besides the previously discussed displacement accuracy, two other important properties of the predictor-corrector algorithms have to be investigated. The first property is that the commanded actuator displacement paths are sufficiently smooth, which becomes important in continuous testing. Sufficiently smooth refers to displacement commands that are C_0-continuous; thus, there is no sudden displacement jumps. While it might seem advantageous to ask for C_1-continuity (continuous first derivatives) as well, it was found that it makes the system more likely to become unstable when switching from prediction to correction. This is because the accuracies of the corrector decrease with increasing continuity at the switching point. The second property that should be investigated is the velocity accuracy, which becomes important in real-time testing. This is especially true if velocity-dependent components such as base-isolation devices and dampers are part of a hybrid simulation.

Returning to the predictor-corrector algorithms discussed in the previous section it becomes obvious that when the algorithm is required to switch within one controller time step, Δt_{con}, (typically one millisecond) from the last predicted displacement value to the first corrected displacement value, a very high-velocity demand is created, resulting in an almost step-like discontinuity. Even though this problem decreases with an increasing order of the Lagrange polynomials, it might still create undesired actuator oscillations (similar to a step input) if the servo-hydraulic control system is tuned to behave highly responsive. Therefore, with the existing extrapolation-interpolation schemes neither C_0-continuity nor velocity accuracy is achieved.

Correction (Cn):

To correct the first property and thereby make the predictor-corrector algorithms C_0-continuous, the corrector Lagrange polynomials are modified to include the last predicted displacement. The

predictor formulas remain unchanged. By replacing the last trial displacement at time step $t_{int,i}$ with the last predicted value, the following expression is obtained for the predictor.

$$dsp_c(x) = \sum_{\substack{k=-n+1 \\ k\neq 0}}^{1} f_k\, C_{n,k}(x, x_p) + f_{x_p}\, C_{n,x_p}(x, x_p) \tag{5.9}$$

where f_k are the displacements calculated by the integrator and f_{x_p} is the last predicted displacement. The range for x is again $x \in [0, 1]$ and $x_p \in [0, 0.8]$. The modified Lagrange functions up to a third-order predictor are given below.

Table 5.3 Modified corrector Lagrange functions up to order 3.

$C_{1,1} = -\dfrac{x - x_p}{x_p - 1}$	$C_{2,1} = -\dfrac{(x - x_p)(x+1)}{2(x_p - 1)}$	$C_{3,1} = -\dfrac{(x - x_p)(x+1)(x+2)}{6(x_p - 1)}$
$C_{1,x_p} = \dfrac{x-1}{x_p - 1}$	$C_{2,x_p} = \dfrac{(x-1)(x+1)}{(x_p - 1)(x_p + 1)}$	$C_{3,x_p} = \dfrac{(x-1)(x+1)(x+2)}{(x_p - 1)(x_p + 1)(x_p + 2)}$
-	$C_{2,-1} = \dfrac{(x-1)(x - x_p)}{2(x_p + 1)}$	$C_{3,-1} = \dfrac{(x-1)(x - x_p)(x+2)}{2(x_p + 1)}$
-	-	$C_{3,-2} = -\dfrac{(x-1)(x - x_p)(x+1)}{3(x_p + 2)}$

To evaluate the accuracy of the new corrector, a sinusoidal vibration is assumed once again. Substitution of (5.3) and of ((5.4) evaluated at x_p) into (5.9) and trigonometric expansion delivers the following result.

$$dsp_c(x) = A\, R_d \sin(\omega t_{int} + \phi) \tag{5.10}$$

where:

$$R_d = \sqrt{C_S^2 + C_C^2}\ , \quad \phi = \arctan\left(\frac{C_C}{C_S}\right)$$

$$C_S = \sum_{\substack{k=-n+1 \\ k\neq 0}}^{1} \cos\left(\omega \Delta t_{int}\left(x-k\right)\right) C_{n,k}\left(x, x_p\right)$$

$$+ C_{n,x_p}\left(x, x_p\right) \sum_{k=-m}^{0} \cos\left(\omega \Delta t_{int}\left(x_p - k\right)\right) P_{m,k}\left(x_p\right) \qquad (5.11)$$

$$C_C = -\sum_{\substack{k=-n+1 \\ k\neq 0}}^{1} \sin\left(\omega \Delta t_{int}\left(x-k\right)\right) C_{n,k}\left(x, x_p\right)$$

$$- C_{n,x_p}\left(x, x_p\right) \sum_{k=-m}^{0} \sin\left(\omega \Delta t_{int}\left(x_p - k\right)\right) P_{m,k}\left(x_p\right)$$

The graphical evaluation and the comparison against all the other algorithms for different values of x and x_p are discussed in a later section, after the formulas for all the investigated predictor-corrector algorithms have been presented.

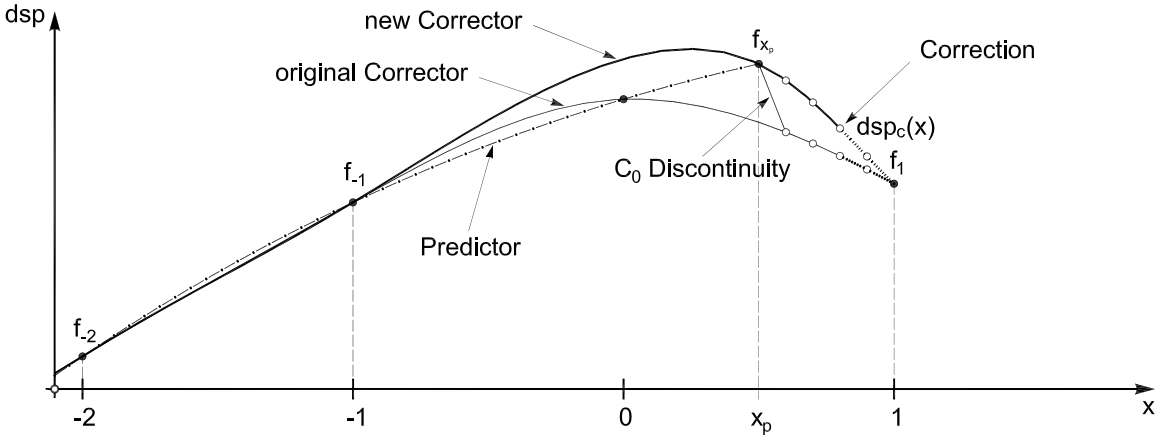


Fig. 5.7 Corrector using a third-order Lagrange polynomial including the last predicted value.

5.3.3 Algorithms Using Velocities

As described in the previous section it would be advantageous not only to achieve C_0-continuity but also to improve the velocity accuracy. While the earlier suggested modifications to the corrector accomplish the sought-after continuity, the velocity accuracy has not been improved yet. A first step towards attaining better velocity accuracy is to incorporate velocities in addition

to the displacements utilized so far. Depending on the employed integration method, accurate trial velocities are readily available and can thus be sent to the predictor-corrector algorithm together with the trial displacements. This is especially true for all the Runge-Kutta methods, where the solution vector directly contains displacements and velocities (see Chapter 4). However, for many of the direct integration methods the trial velocities at the integration time steps or substeps (iterative methods) are not very accurate or unavailable, and they are therefore incapable to improve the velocity accuracy of the predictor-corrector algorithms. In these cases velocity estimates can be obtained from numerical differentiation formulas as shown below.

Prediction (PV):

In this instance, the predicted displacements are expressed in terms of a Hermit polynomial using the last n displacement values as well as the last n velocity values.

$$dsp_p(x) = \sum_{k=-n}^{0} f_k P_{2n+1,k}(x) + \sum_{k=-n}^{0} f_k' \hat{P}_{2n+1,k}(x) \tag{5.12}$$

where f_k and f_k' are the displacements and velocities at the integration time steps, and, as always, $x \in [0, 0.8]$. For convenience the Hermit functions for the third-order polynomial are given below.

$$\begin{aligned} P_{3,0} &= -(x+1)^2(2x-1) & \hat{P}_{3,0} &= (x+1)^2 x \\ P_{3,-1} &= x^2(2x+3) & \hat{P}_{3,-1} &= x^2(x+1) \end{aligned} \tag{5.13}$$

If the velocities f_k' in (5.12) are received from the integration method, they have to be multiplied by the integration time step, Δt_{int}, before the substitution, since the x-axis is dimensionless and not in units of time.

On the other hand, if the velocities are determined utilizing numerical differentiation formulas, they can directly be substituted into (5.12) because the x-axis is already dimensionless. The differentiation formulas for second-order accurate velocities take the following form.

$$\begin{aligned} f_0' &= \frac{1}{2}(3f_0 - 4f_{-1} + f_{-2}) \\ f_{-1}' &= \frac{1}{2}(f_0 - f_{-2}) \end{aligned} \tag{5.14}$$

For third-order accuracy the velocities are expressed in the following manner.

$$f_0' = \frac{1}{6}(11f_0 - 18f_{-1} + 9f_{-2} - 2f_{-3})$$
$$f_{-1}' = \frac{1}{6}(2f_0 + 3f_{-1} - 6f_{-2} + f_{-3}) \tag{5.15}$$

And finally the fourth-order formulas approximate the velocities as follows.

$$f_0' = \frac{1}{12}(25f_0 - 48f_{-1} + 36f_{-2} - 16f_{-3} + 3f_{-4})$$
$$f_{-1}' = \frac{1}{12}(3f_0 + 10f_{-1} - 18f_{-2} + 6f_{-3} - f_{-4}) \tag{5.16}$$

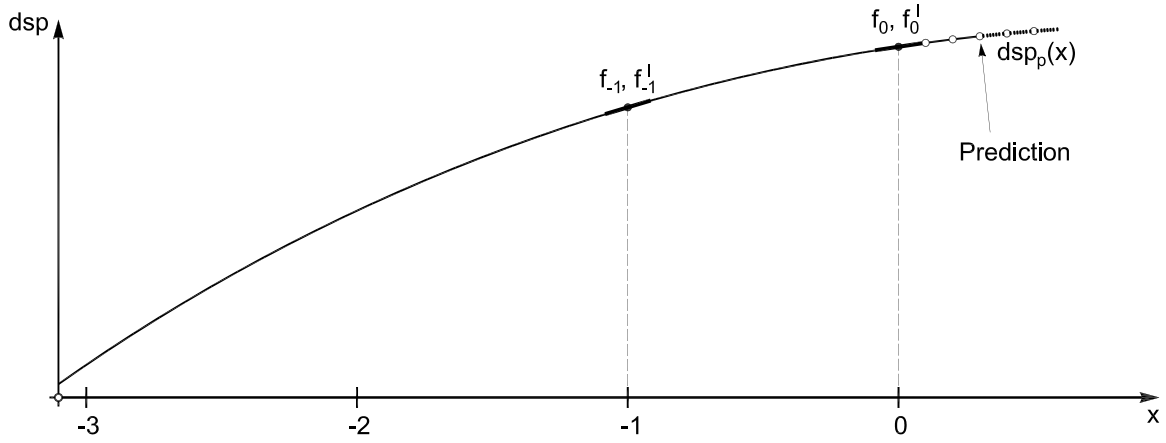


Fig. 5.8 Predictor using a third-order Hermit polynomial.

The accuracy of this predictor is evaluated in the same manner as for the previous ones except that the second-, third-, or fourth-order velocities are substituted into (5.12) before the amplitude change R_d and the phase shift ϕ are determined. Due to their complexity these formulas are no longer presented here, but the results are shown graphically at the end of this section.

Correction (CV):

In order for a corrector formula to increase the velocity accuracy while maintaining the C_0-continuity, trial velocities as well as the last predicted displacement value need to be incorporated. Again, as for the previously suggested predictor, only C_0-continuity is provided in the switching interval. This makes the system more stable in case the predicted displacements are fairly inaccurate. Compared to the almost jump-like displacement command that was encountered in the switching interval of the original algorithm, the C_0-continuity permits a discontinuity only in the velocity. This reduces possible oscillations significantly. Furthermore,

since C_1-continuity is not enforced, the corrector does not use the erroneous last velocity of the predictor to initially shoot in the wrong direction before correcting to the next trial displacement.

$$dsp_c(x) = \sum_{\substack{k=-n+1 \\ k\neq 0}}^{1} f_k\, C_{2n,k}(x, x_p) + \sum_{\substack{k=-n+1 \\ k\neq 0}}^{1} f_k'\, \hat{C}_{2n,k}(x, x_p) + f_{x_p}\, C_{2n,x_p}(x, x_p) \tag{5.17}$$

where f_k and f_k' are the displacements and velocities at the integration time steps and f_{x_p} is the last predicted displacement. The ranges for x and x_p are as before $x \in [0, 1]$ and $x_p \in [0, 0.8]$. As with the predictor formula the velocities f_k' in (5.17) have to be multiplied by the integration time step, Δt_{int}, if they are received from the integration method. Since this corrector can have only order $2n$, the second-order functions are given as an example below. The second-order corrector is preferred over the fourth-order one because less time and memory are consumed to calculate the command displacements.

$$C_{2,1} = -\frac{(x + x_p - 2)(x - x_p)}{(x_p - 1)^2} \qquad \hat{C}_{2,1} = -\frac{(x-1)(x - x_p)}{x_p - 1}$$
$$C_{2,x_p} = \frac{(x-1)^2}{(x_p - 1)^2} \tag{5.18}$$

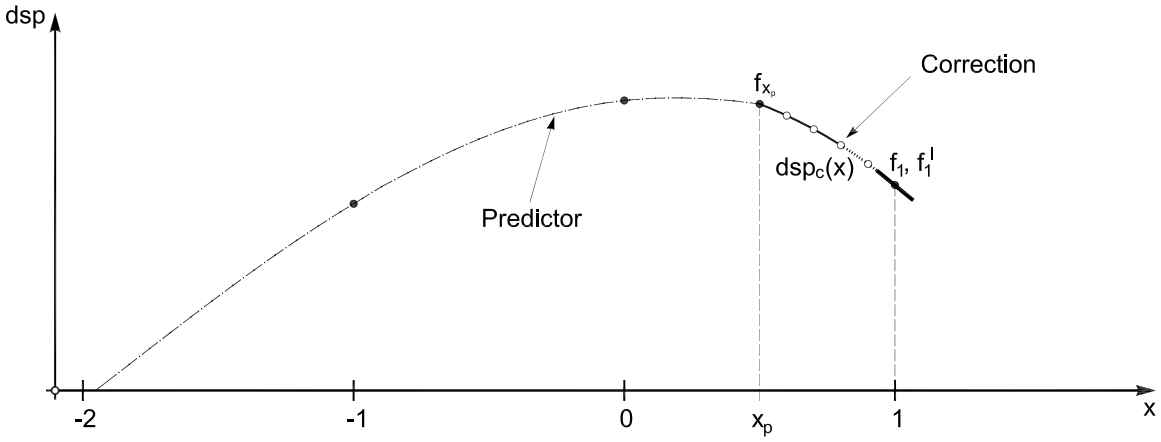


Fig. 5.9 Corrector using a second-order polynomial including the last predicted value.

Once more the accuracy formulas are not presented here due to their complexity, but the results are shown graphically at the end of this section.

5.3.4 Algorithms Using Accelerations

So far, all the predictor-corrector algorithms were based on polynomial approximations used in numerical mathematics and less so on the physical behavior of a system.

Prediction (PVA):

In order to further improve the prediction and correction of actuator command signals, the physical behavior of a mass under the action of a constant force is incorporated into the algorithm. This means that it is assumed that the force does not change over a time step and is equal to the force at the beginning of the step. If the sum of all acting forces (applied, damping, and resisting) is constant, then by Newton's second law of motion, the acceleration has to be constant over a time step as well. Integrating the constant acceleration twice and using the displacement and velocity at the beginning of the time step as initial conditions, the following equation for the predictor is obtained.

$$dsp_p(x) = f_0 + x f_0' + \frac{1}{2} x^2 f_0'' \tag{5.19}$$

where f_0, f_0', and f_0'' are the displacement, velocity, and acceleration, respectively, at the last integration time step, and $x \in [0, 0.8]$ as always. If the velocities f_0' and accelerations f_0'' in (5.19) are received from the integration method, they have to be multiplied by Δt_{int} and Δt_{int}^2, respectively, since the x-axis is dimensionless and not in units of time.

Alternatively, if the velocities and accelerations are determined utilizing numerical differentiation formulas, they can be substituted directly into (5.19) because the x-axis is already dimensionless.

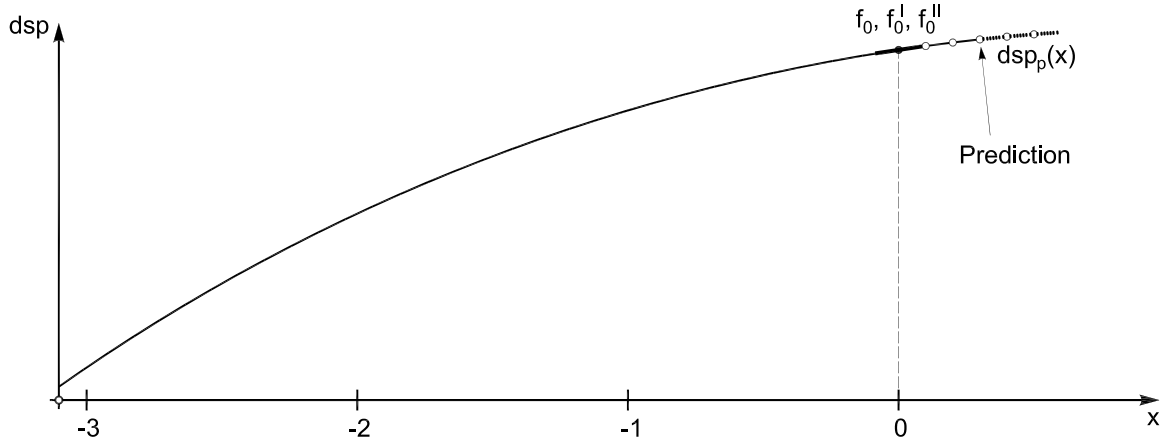


Fig. 5.10 Constant acceleration predictor.

The differentiation formulas for the velocities remain unchanged from the previous subsection. The formulas for the second- to fourth-order accurate accelerations take the following forms.

$$f_0'' = 2f_0 - 5f_{-1} + 4f_{-2} - f_{-3} \tag{5.20}$$

$$f_0'' = \frac{1}{12}\left(35f_0 - 104f_{-1} + 114f_{-2} - 56f_{-3} + 11f_{-4}\right) \tag{5.21}$$

$$f_0'' = \frac{1}{12}\left(45f_0 - 154f_{-1} + 214f_{-2} - 156f_{-3} + 61f_{-4} - 10f_{-5}\right) \tag{5.22}$$

The accuracy of this predictor is evaluated in the same manner as for the previous ones except that the second-, third-, or fourth-order velocities and accelerations are substituted into (5.19) before the amplitude change R_d and the phase shift ϕ are determined. Due to their complexity these formulas are again omitted here, but the results are shown graphically at the end of this section.

Correction (CVA):

The expression for a corrector that is making use of accelerations is derived similarly to the previous one, which used only displacements and velocities. The only differences between the two are the addition of the third term and the change of the order.

$$\begin{aligned} dsp_c(x) = &\sum_{\substack{k=-n+1 \\ k\neq 0}}^{1} f_k\, C_{3n,k}\left(x, x_p\right) + \sum_{\substack{k=-n+1 \\ k\neq 0}}^{1} f_k'\, \hat{C}_{3n,k}\left(x, x_p\right) \\ &+ \sum_{\substack{k=-n+1 \\ k\neq 0}}^{1} f_k''\, \tilde{C}_{3n,k}\left(x, x_p\right) + f_{x_p}\, C_{3n,x_p}\left(x, x_p\right) \end{aligned} \tag{5.23}$$

where f_k, f_k' and f_k'' are the displacements, velocities, and accelerations, respectively, at the integration time steps, and f_{x_p} is the last predicted displacement. The ranges for x and x_p are as before, $x \in [0, 1]$ and $x_p \in [0, 0.8]$. The functions that multiply the displacements, velocity and acceleration for a third-order corrector are given by.

$$C_{3,1} = -\frac{\left(x^2 + \left(x_p - 3\right)x + x_p^2 - 3x_p + 3\right)\left(x - x_p\right)}{\left(x_p - 1\right)^3} \qquad \hat{C}_{3,1} = -\frac{\left(x + x_p - 2\right)\left(x - x_p\right)\left(x - 1\right)}{\left(x_p - 1\right)^2}$$

$$\tilde{C}_{3,1} = -\frac{\left(x-1\right)^2\left(x - x_p\right)}{2\left(x_p - 1\right)} \qquad C_{3,x_p} = \frac{\left(x-1\right)^3}{\left(x_p - 1\right)^3} \tag{5-24}$$

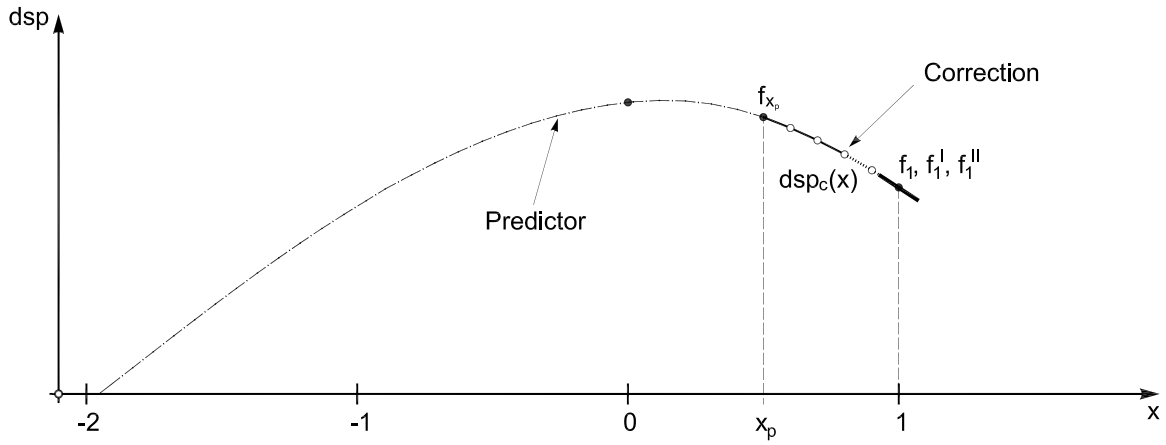


Fig. 5.11 Corrector using a third-order polynomial including the last predicted value.

5.3.5 Graphical Evaluation and Comparison

In order to evaluate the different predictor-corrector algorithms graphically and compare them, the calculated amplitude and phase accuracies are plotted against the normalized integration time step, $\Delta t_{int}/T$. While for the low-frequency modes of vibration, the normalized integration time step will be on the lower end of the plotted range, it can lie in the upper part of the plotted range for the higher modes of a multi-degrees-of-freedom system. If these high-frequency modes contribute significantly to the overall response of the structure, their amplitude and phase accuracies are essential as well. The amplitude change R_d and phase shift ϕ are plotted for three different values of x_p (0.2, 0.5 and 0.8). For the predictors, $x = x_p$ and $x \in [0, 0.8]$. As mentioned earlier, the prediction is limited to a maximum of 80% of the simulation time step in order to leave a minimum of 20% of the simulation time step for the correction towards the next trial displacement. For the correctors $x = 0.5 + x_p/2$ is chosen to evaluate the accuracy in the middle between the last predicted value and the new trial value and the fourth-order numerical differentiation formulas are utilized to obtain velocities and accelerations.

Predictors:

As can be seen from Fig. 5.12, the amplitude and phase accuracies of the two newly proposed predictors are overall better than the accuracies of the original ones. In addition, the accuracies of the newly developed predictors are generally very similar where the one using accelerations is slightly more accurate than the other using velocities only. It can be seen that the new predictors are significantly more accurate for the low-frequency modes ($\Delta t_{int}/T < 0.2$). Above such value their accuracy is less consistent and starts picking up some oscillatory behavior due to the high-order polynomials that are generated by the numerical differentiation formulas. This fact, which is well known from numerical mathematics, is called the Runge phenomenon. It states that by interpolating a function at equidistant points in a given interval using a high-order polynomial, the resulting interpolation will oscillate towards the end of the interval. This is also the reason why the original predictors go up only to order three, because the fourth-order polynomial already shows significant oscillatory behavior and is thus less accurate for the high-frequency modes.

Correctors:

Fig. 5.13 shows the amplitude changes and the phase shifts for the corrector algorithms, which do not incorporate the last predicted displacement. It can be seen that the correctors are overall more accurate than the predictors. This is due to the fact that the trial response quantities at the new step (or substep) are available to the correctors, and the calculated command displacements are therefore within the interval of known values. On the other hand, the calculated command displacements during prediction lie outside the interval of known values. However, it is important to remember that these corrector algorithms are unable to generate C_0-continuous displacement commands and therefore produce a step-like signal in the switching interval. Such inaccuracies and deficiencies are graphically not captured in Fig. 5.13. Furthermore, it can be observed that the correctors that are incorporating velocities and accelerations achieve about the same order of accuracy as the original second- and third-order correctors. It is also interesting to note that the corrector that incorporates velocities is overall the most accurate one.

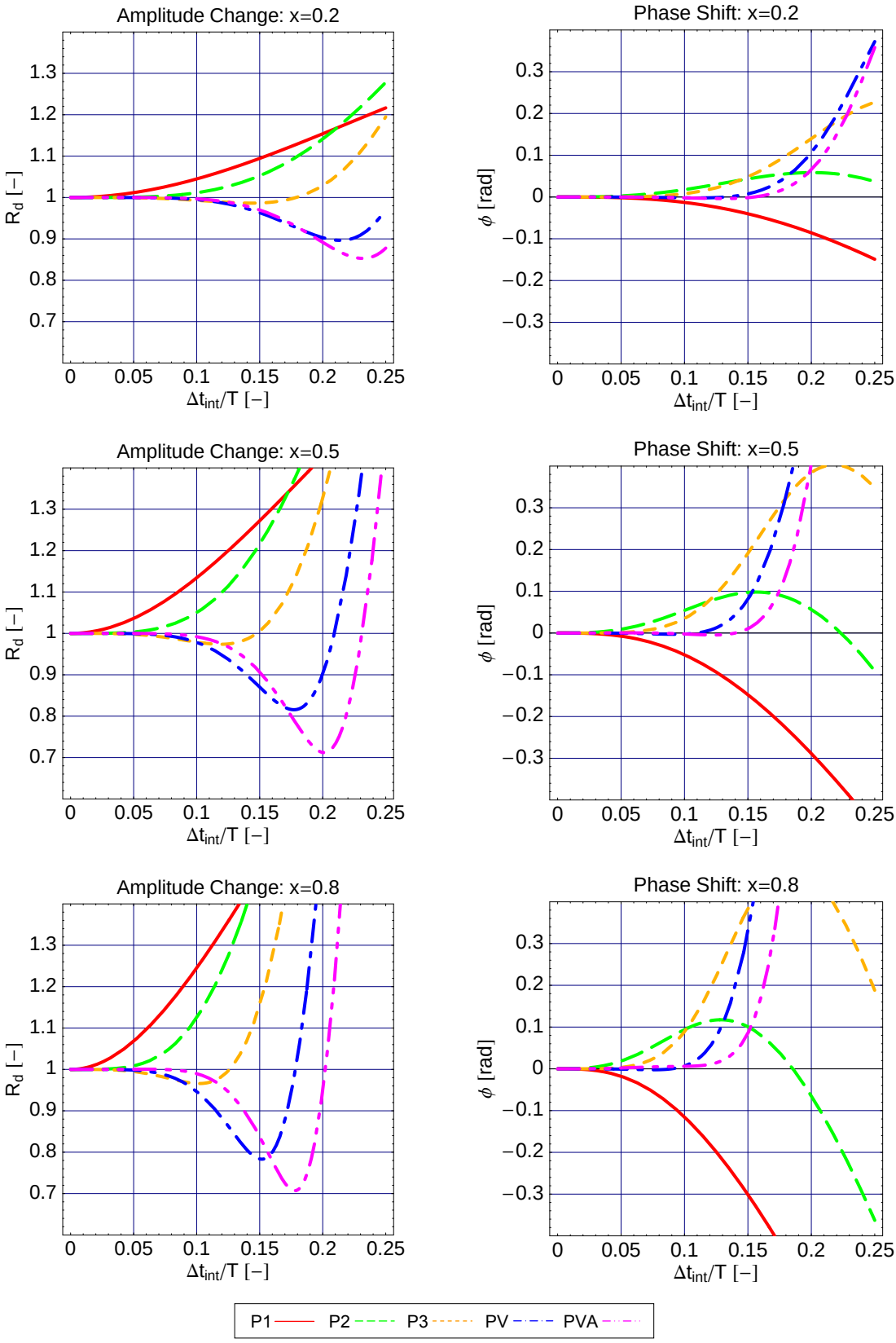


Fig. 5.12 Accuracies of predictors in terms of amplitude and phase.

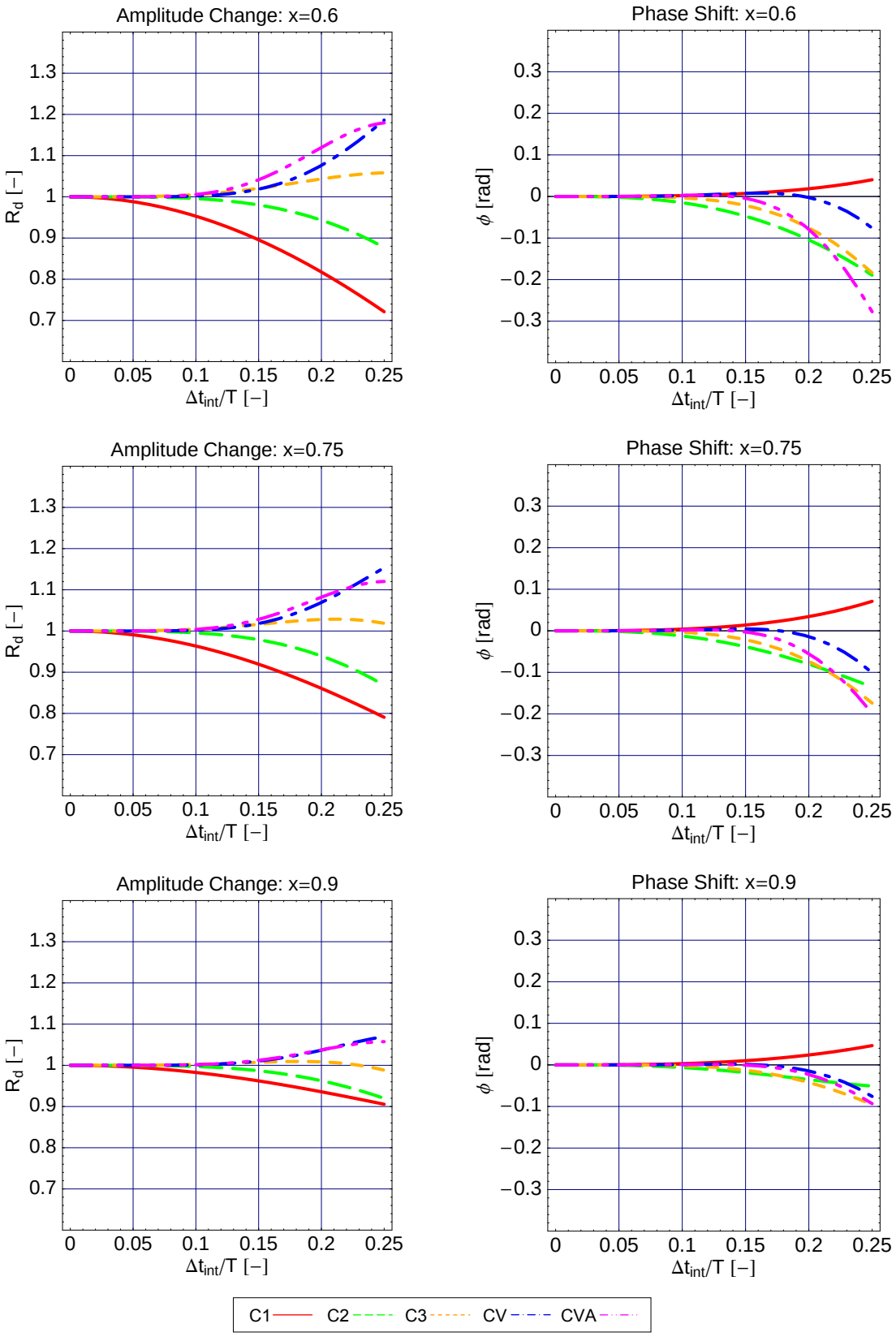


Fig. 5.13 Accuracies of correctors in terms of amplitude and phase.

Correctors Using Last Predicted Displacement:

It is evident that overall the C_0-continuous predictors are slightly less accurate than their discontinuous counterparts. This effect is due to the incorporation of the last predicted displacement into the corrector formulas, which even at $x_p = 0.2$ already have some prediction errors. However, at the cost of slightly reduced accuracies the step-like discontinuities in the switching interval are entirely eliminated. From Fig. 5.14, it can be seen that this trend for the amplitude and phase accuracies becomes more pronounced the larger x_p gets. The further away the prediction is from the last known value, the larger the error in the last predicted displacement and the more inaccurate the C_0-continuous correctors perform. However, one has to keep in mind that for the discontinuous correctors, the jump in the switching interval increases as well with increasing x_p. And, if this almost step-response-like switching discontinuity becomes too large, a responsive servo-hydraulic system will introduce overshoot and oscillations into the actuator commands at those points making the correction very inaccurate. So, it seems that a C_0-continuous corrector with a somewhat reduced accuracy would be preferable over a discontinuous corrector that might initiate oscillations into the actuator commands. As can be seen from Fig. 5.14, for small values of x_p all the corrector algorithms achieve quite similar orders of accuracy, with the C3 and CV correctors being the most accurate. On the other hand, for larger values of x_p it is evident that the correctors that incorporate velocities and accelerations achieve better performance, with the CVA corrector clearly being the most accurate and thus preferred algorithm.

5.3.6 Advantages, Disadvantages, and Recommendations

While the advantages and disadvantages of each individual predictor and corrector are summarized in Table 5.4 below, some general advantages and disadvantage are first introduced. In the hybrid simulation of complicated structures that consist of nonlinear elements in their analytical subassemblies, it cannot always be guaranteed that the solution of the equations of motion is available to the controller (which is commanding the transfer system) on time. This is even true if an explicit direct integration method is used to solve the equations of motion. For geographically distributed hybrid simulations among several laboratories, this problem becomes even more serious because the solutions will inevitably be delayed due to latencies in the network communications.

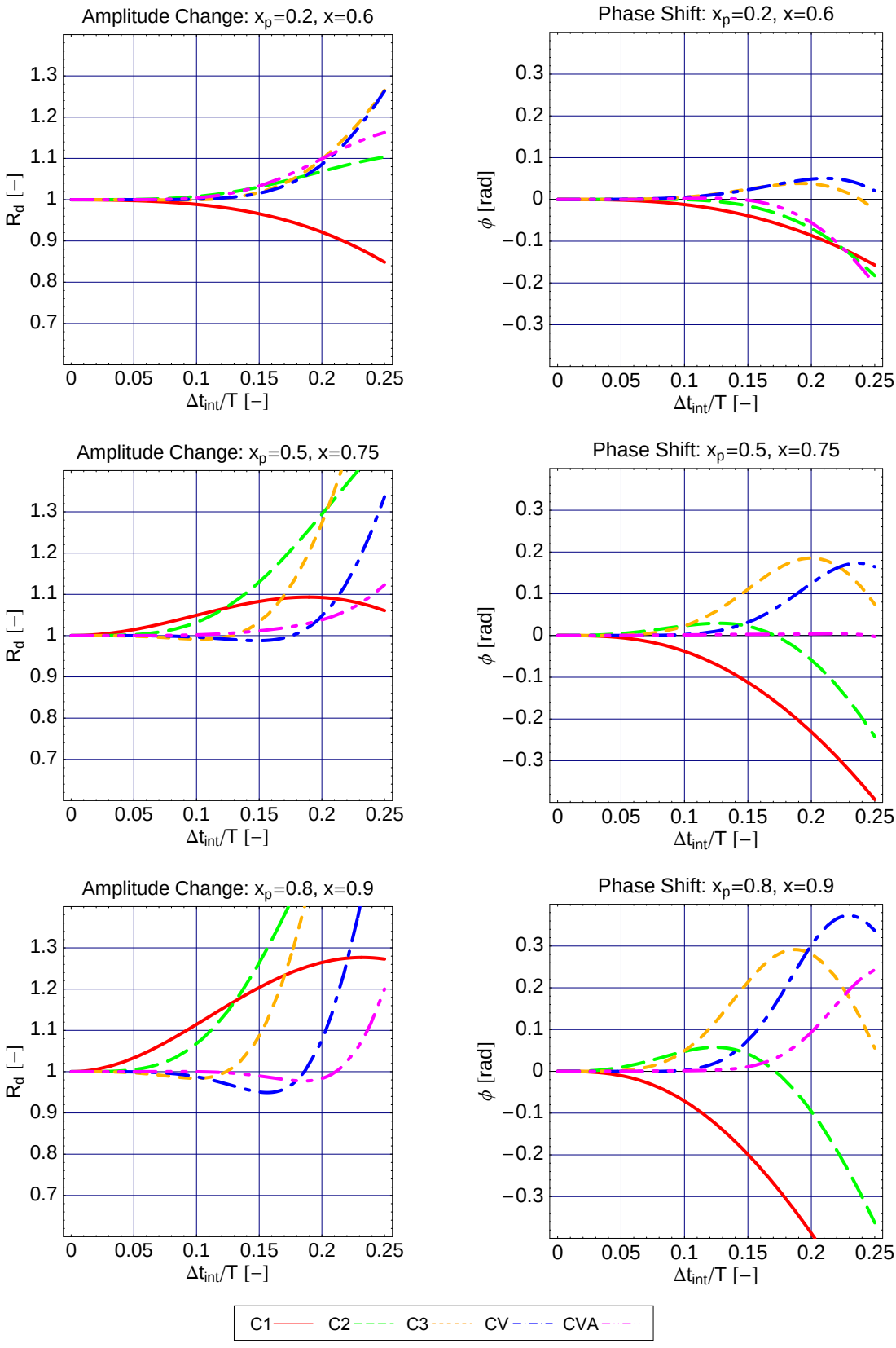


Fig. 5.14 Accuracies of correctors including last predicted displacement.

Thus, the main advantage of the predictor-corrector algorithms is that they eliminate this problem by continuously producing actuator command signals, thereby creating a bridge between the computational driver and the transfer system imposing the boundary conditions. Another advantage comes directly from the ability to produce continuous actuator command signals. This eliminates or at least reduces the problems associated with force relaxation. On the other hand, the biggest problem with the predictor-corrector algorithms is that they need to predict the response at certain degrees of freedom of a structure without any knowledge about the properties of such structure. Hence, even though the predicted response is continuous, it could be very erroneous. Another problem is that in their current form the predictor-corrector algorithms do not work well with iterative integration methods. This problem and, accordingly, solution strategies are addressed in a later section of this chapter.

Table 5.4 Summary of advantages/disadvantages for predictor-corrector algorithms.

Algorithm	*Advantages*	*Disadvantages*
Hold and Ramp	○ Works well with iterative integrators ○ Fast calculation due to easy formulas	○ Discontinuous actuator motions with noniterative as well as iterative integrators ○ Very inaccurate compared to continuous command signals ○ Force relaxation problems during hold phase ○ Potentially high velocities in ramp phase if hold phase takes up 80% of the time step
Original Dis-continuous	○ Fairly accurate, low-order correctors	○ Based on pure numerical polynomial interpolation ○ Considerable inaccuracies for high-frequency modes or large integration time steps ○ Step-like discontinuity in switching interval which can cause oscillations of servo-hydraulic control system ○ Oscillations of high-order polynomials (Runge-phenomenon) ○ Very inaccurate if used with iterative integrators
Modified C_0-Continuous	○ C_0-continuous and therefore no step-like discontinuity in the switching interval	○ Based on pure numerical polynomial interpolation ○ Large inaccuracies for high-

		frequency modes or large integration time steps o Oscillations of high-order polynomials (Runge-phenomenon) o Very inaccurate if used with iterative integrators
Utilizing Velocities	o Predictor and corrector achieve good accuracy o C_0-continuous	o Based on pure numerical polynomial interpolation
Utilizing Accelerations	o Based on the physical behavior of a mass o Most accurate of all the methods o C_0-continuous	o Minor oscillations at larger values of the normalized integration time step (slight Runge-phenomenon)

To conclude, it is recommended that the predictor-corrector algorithms, which are based on velocities and accelerations and are C_0-continuous, are utilized whenever possible. This is especially true if noniterative integration methods or integration schemes with constant numbers of iterations are employed (see Chapter 4). Furthermore, all the discontinuous predictor-corrector algorithms should be avoided and their C_0-continuous counterparts should be utilized instead. This eliminates step-like command input conditions in the switching interval and as a consequence, reduces displacement and thus force oscillations that often destabilize transfer systems. On the other hand, if unmodified iterative integration schemes are employed in the finite element analysis software, the traditional hold and ramp algorithms or the adaptive methods that are discussed later should be utilized to avoid displacement oscillations.

5.4 ADAPTIVE EVENT-DRIVEN STRATEGIES

5.4.1 Existing Strategy

After the development and discussion of two of the key elements of the three-loop architecture, the next step is to investigate how the improved synchronization predictor-corrector algorithms can be incorporated into the intermediate loop of such architecture. As mentioned earlier, the time required to advance the solution of the equations of motion by the integration time step, Δt_{int}, is not constant. This is especially true for complex structures with material and geometric nonlinearities in their analytical subassemblies. In geographically distributed hybrid simulations,

the time that passes until the next trial response quantities are available happens to be even less predictable. Random delays due to network latencies have to be added to the time consumed by the computational driver. To deal with this randomness of the system, Mosqueda (2003) introduced an event-driven algorithm (a finite-state machine) that utilizes the predictor-corrector algorithms and is placed in the intermediate loop of the three-loop architecture. A finite-state machine is composed of states in which the system can exist and transition between those states. Guards and/or triggers (i.e., events) control the execution of the transitions. Furthermore, actions can be attached to transitions and/or states.

The existing event-driven controller (Mosqueda 2003) is composed of four states. During standard operation, the trial response quantities from the integration method are available without delays and the controller will simply switch between the *Prediction* state and the *Correction* state. The actions that are executed while these states are active are the third-order extrapolation and the third-order interpolation methods of the predictor-corrector algorithms. However, if the integration method is taking too long to calculate the new response quantities or the transfer of such trial quantities across the network is delayed, the algorithm adapts and slows down the prediction. This is achieved by triggering a transition from the *Prediction* state into a *SlowDown* state. The guard on this transition is checking that a given percentage of the simulation time step, Δt_{sim}, has passed in order to execute the transition. The actions, which are executed while in the *SlowDown* state, are still third-order extrapolations but they move the actuators at half the speed, thus allowing more time to receive the next trial response quantities. If the new prescribed boundary conditions become available within a specified time frame, the finite-state machine is triggered to transition from the *SlowDown* state back to the *Correction* state. On the other hand, if the trial response quantities have still not been received after the specified time has passed, the controller transitions into a *Hold* state. During the *Hold* state, constant command signals are sent to the control system, which effectively stops the actuators from moving. Once the response quantities at the next time-step become available, the controller is triggered to transition back to the *Correction* state. Nakashima and Masaoka (1999) found that the most appropriate values to use for the two guards between the *Prediction*, *SlowDown,* and *Hold* states are 60% and 80% of the simulation time step, Δt_{sim}. This means that after 60% of the simulation time step has passed, the controller slows the actuator movements down to half the

speed and completely stops the movements at 80% of the simulation time step. This leaves a guaranteed minimum of 20% of the simulation time step for subsequent corrections.

5.4.2 Smooth Slow-Down Strategy

The problem that arises with the just-described event-driven strategy is that there is no smooth transition from the *Prediction* state to the *Hold* state and from the *Hold* state back to the *Correction* state. The velocities are not continuous, since the transfer system jumps from full speed to half speed to zero speed and back to full speed. If these velocity discontinuities are large, they can cause oscillations in the actuator movements as they were encountered in real tests in the laboratory. The response of the actuators to those velocity discontinuities depends a lot on all the tuning parameters of the control system. However, if these discontinuities are already eliminated in the implementation of the event-driven strategy, then the problem of tuning the servo-hydraulic control system to be responsive enough while trying to reduce oscillations can be avoided.

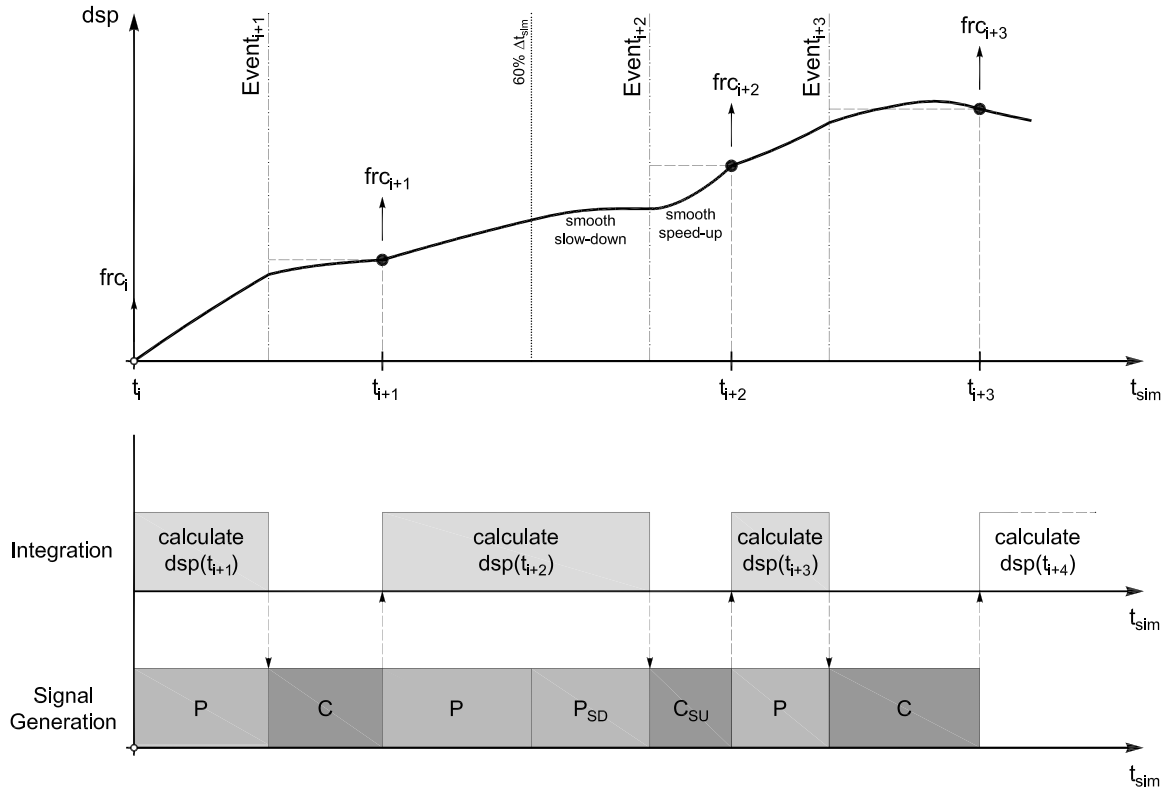


Fig. 5.15 Displacement path and execution sequence of smooth slow-down strategy.

In order to improve the event-driven controller, a continuous change of the actuator velocities is incorporated into the *SlowDown* and the *Correction* states. Since the *SlowDown* state reduces the actuator velocities smoothly all the way to zero, the previously used *Hold* state is superfluous and is therefore eliminated. The *Correction* state is modified such that if it is entered from the *SlowDown* state, it increases the actuator velocities smoothly from the last velocities in the *SlowDown* state back to the full velocities at the target. In contrast, if the *Correction* state is entered directly from the *Prediction* state during standard operation, no velocity changes take place. These two modes of operation are illustrated in Fig. 5.15 below.

To obtain the sought after smooth velocity change in the *SlowDown* state, the counter-increment is linearly reduced from $\Delta_{i,SD}=1$ at 60% of the integration time step to $\Delta_{i,SD}=0$ at 80% of the integration time step. The reduction of the counter-increment effectively slows down the actuators because additional substeps (each one of length Δt_{con}) are being generated within the simulation time step. Similarly, the counter-increment is linearly increased again during the *Correction* state starting from the last used counter-increment in the *SlowDown* state $\Delta_i=\Delta_{i,SD}$ back to $\Delta_i=1$.

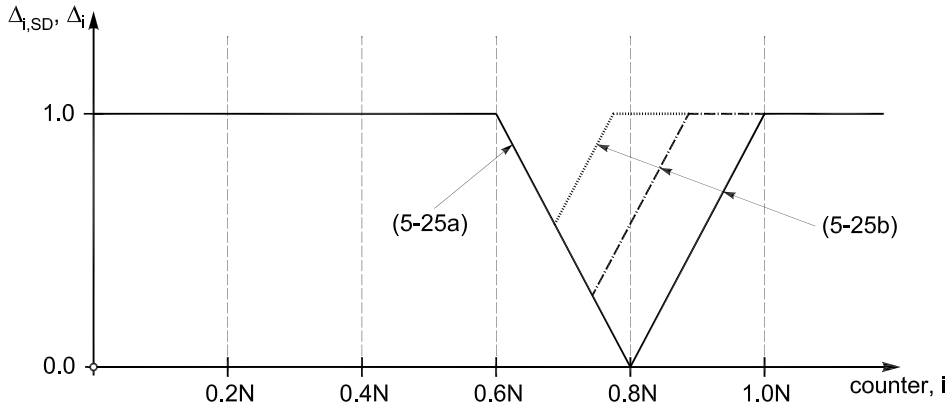


Fig. 5.16 Relationship between counter-increment and counter for smooth velocity transitions.

$$\Delta_{i,SD}=1.0-\frac{1.0}{0.2N}(i-0.6N)=4.0-\frac{i}{0.2N}$$
$$\Delta_i=\min\left[\max\left[\Delta_{i,SD}+\frac{1.0}{0.2N}(i-i_{SD}),0.001\right],1.0\right] \tag{5.25}$$

In the above formulas, i is the counter and N is the total number of substeps (which is equal to the simulation time step Δt_{sim} divided by the controller time step Δt_{con}). In formula

(5.25) the maximum condition ensures that the velocity can increase again if it dropped all the way to zero and the minimum condition guarantees that the counter increment does not increase above one. All these event-driven strategies are implemented in Stateflow, a graphical design and development toolbox in Matlab/Simulink (Mathworks) for simulating complex reactive systems based on finite state machine theory. On the other hand, the previously discussed predictor-corrector functions are implemented using the C programming language. By including the header file and specifying the source file within Stateflow, access to the whole library of discussed predictor-corrector algorithms is obtained. These event-driven strategies are then incorporated into a Simulink (Mathworks) model, which runs on a real-time platform and communicates with the finite element analysis software utilizing the OpenFresco middleware (see Chapter 3).

5.4.3 Adaptive Velocity Strategies

Several situations arise in hybrid simulation where it becomes advantageous to automatically adjust actuator velocities and thus testing speeds. Often times when executing a hybrid simulation of a structure excited by an earthquake, the interesting parts of the experimental test are strongly correlated with the intensity of the ground motion. Meaning, that the beginning and the end of a test are usually less interesting due to the fact that the ground motion intensity is very low and the structural response is very small. As long as the experimental subassemblies do not exhibit rate-dependent behaviors, one possible solution to this problem is to execute a hybrid simulation with constant, user-specified actuator velocities. Since the actuators move at a constant velocity, testing is effectively accelerated for those parts of the simulation that produce small structural responses and is thus made more interesting. To achieve this goal the simulation time step Δt_{sim} (respectively the total number of substeps N) is adjusted during simulation according to the size of the trial response increment received from the finite element analysis software.

$$N = \text{ceil}\left(\frac{|f_0 - f_{-1}|}{v_{Act}\,\Delta t_{con}}\right) \tag{5.26}$$

where f_0 and f_{-1} are the trial displacements at the new and previous integration steps, v_{Act} is the user-specified actuator velocity, and Δt_{con} is the time step of the control system. If multiple actuators are commanded by a single control system, one of them is chosen as the

master actuator for which the user specifies the desired velocity. The remaining actuators are then slaved to the master in the sense that their velocities are given, because they depend on the total number of substeps N, which is determined from the master. To prevent the actuators from exceeding their velocity limits and to guarantee that N does not fall below a minimum number of substeps, additional checks are performed after the evaluation of (5.26).

A second situation where it is critical to adapt actuator speeds during the course of a hybrid simulation arises when iterative integration methods are employed. As was explained earlier in Chapter 4, displacement increments decrease rapidly during the iteration process due to the quadratic convergence of the Newton-Raphson algorithm. If the control system is allotted the same time interval to impose such inconsistent displacement increments, velocity and thus force oscillations are generated in the transfer system. Hence, to lessen these oscillations, the event-driven strategy needs to adjust the time interval over which the displacement increments are imposed during simulation. To achieve this goal the size of the response increments are checked as they are received from the finite element analysis; and if they exceed a user-specified limit ΔU_{lim} the time over which such increments are applied (respectively the total number of controller substeps N) is automatically increased.

$$\begin{aligned} |f_0 - f_{-1}| &\le \Delta U_{\text{lim}} \quad \rightarrow reset\ N \\ |f_0 - f_{-1}| &> \Delta U_{\text{lim}} \quad \rightarrow increase\ N \end{aligned} \tag{5.27}$$

This means that the actuators are effectively slowed down when they impose the larger displacement increments at the beginning of the iteration process and move at the normal rate while applying the smaller increments towards the end of the equilibrium solution process.

5.4.4 Interlaced Solution Strategy

In the interlaced solution strategy two synchronized integration methods are interlaced in time, meaning that both schemes have an integration time step equal to $2\Delta t_{int}$, but they are spaced at half of this time step. Furthermore, it is assumed that both integration methods run on the same machine (in the outermost loop of the three-loop architecture), while a predictor-corrector algorithm runs on a separate real-time platform in the intermediate loop as before.

According to Fig. 5.17, the interlaced solution procedure has the following steps: Once the trial displacements calculated by integrator A have been reached at $t_{a,i}$, the corresponding

resisting forces $frc_{a,i}$ are measured (this is indicated by the upwards pointing arrow). Integrator A then uses such resisting forces to compute the trial displacements at the next time step $t_{a,i+1}$. In the meanwhile, the corrector algorithm is generating displacement command signals shooting towards the known trial displacements at time $t_{b,i}$. Once these trial displacements at $t_{b,i}$ are reached, the corresponding resisting forces are measured and integrator B starts to solve the equations of motion for the new response at $t_{b,i+1}$. This sequence of operations is repeated for the total number of time steps divided by two. It can be seen from Fig. 5.17, as well as concluded from the assumptions above, that under normal operation the maximum allotted computation time per time step is equal to Δt_{sim} for both of the integration methods.

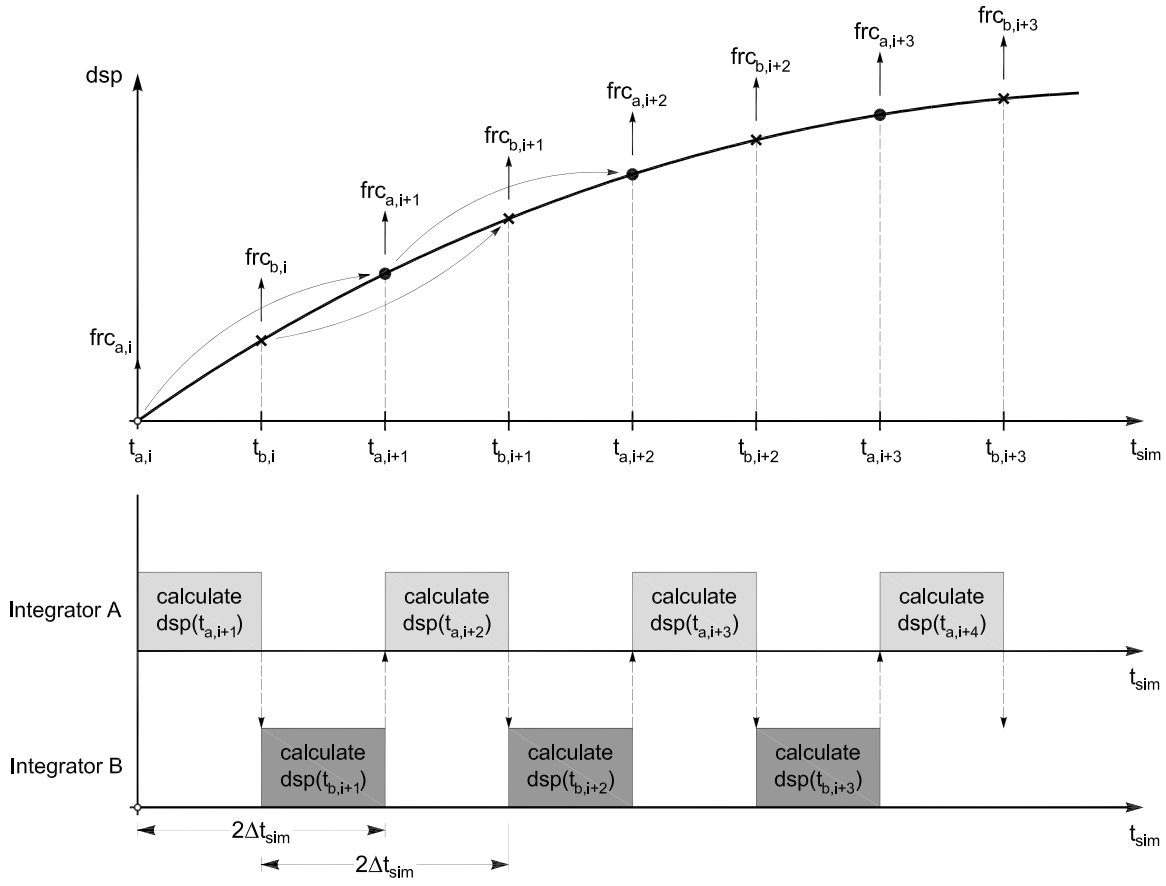


Fig. 5.17 Displacement path and execution sequence of interlaced solution strategy.

The big advantage of the interlaced solution strategy is that the machine that is solving the equations of motion is working to full capacity. This means that contrary to solution strategies with a single integrator, there is no idle time between computation tasks. Such idle times in single integrator strategies are inevitable. This is because after the calculated trial

displacements are sent off, the integrator has to wait until the displacements are applied and the corresponding forces have been measured (see Fig. 5.15). Because the computer that is executing the integration task is working to full capacity, the hybrid simulation can be run at a faster speed. This means that a shorter simulation time step Δt_{sim} can be chosen than in the case where only one integrator is engaged. To cope with the randomness of network communication times in geographically distributed hybrid simulations, the previously developed event-driven predictor-corrector algorithms can be employed again. As before, such algorithms will slow down the displacement commands at 60% of the simulation time step, and hold them at 80% of the simulation time step. In addition, the advantage remains that the integrator machine is working to full capacity and is therefore producing trial displacements faster than a machine running a single integrator.

Because the two integration schemes in the interlaced solution strategy both utilize an integration time step that is equal to $2\Delta t_{int}$, some further attention needs to be paid to the time-step selection. To investigate the continuity and uniformity of the displacement command signals that are imposed on the experimental portions of a structure, a two-degrees-of-freedom-of-freedom, one-bay-frame model is analyzed using the interlaced solution strategy. The one-bay-frame model is equivalent to the one that was utilized for the rapid geographically distributed hybrid simulations (see Chapter 6). It is defined by two column elements that are connected by a fairly flexible linear-elastic truss element. While the right linear-elastic column is always modeled analytically, the left column is tested experimentally and exhibits either a linear or nonlinear behavior. The explicit Newmark method was employed for both of the interlaced solution schemes with integration time-step sizes of $\Delta t_{int} = 0.02$ and $\Delta t_{int} = 0.01$ sec. For comparison, the hybrid model was also analyzed by the regular explicit Newmark integration method with a time-step size of $\Delta t_{int} = 0.02$. Fig. 5.18 shows the entire time history as well as close-ups of the displacement command signals at the controlled degree of freedom of the experimental element. In the linear case, it can be observed that the interlaced solution strategy generates somewhat jagged displacement commands if the original integration time step of $\Delta t_{int} = 0.02$ sec is utilized. However, by selecting an integration time step that is half as large, $\Delta t_{int} = 0.01$ sec, the interlaced solution strategy is capable of producing a smooth and uniform trial response for the transfer system.

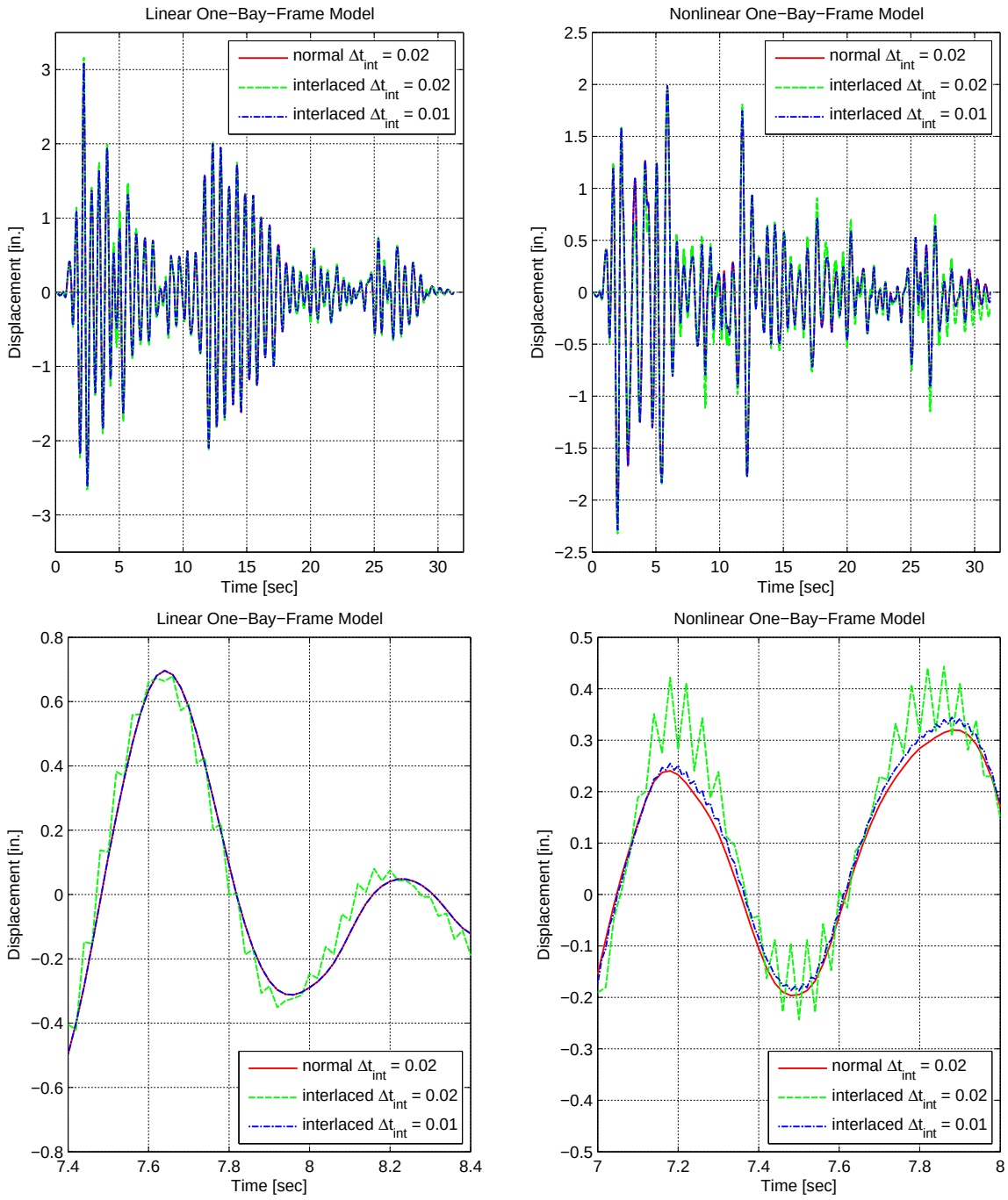


Fig. 5.18 Displacement command signal comparison for interlaced solution strategy.

In the nonlinear case, a similar but more severe behavior can be observed, meaning that the displacement commands are more jagged and inaccurate for large integration time steps. Since the trial response is still somewhat jagged even for $\Delta t_{int} = 0.01$, the integration time-step size should be further reduced in order to generate smooth enough command signals for the transfer system. So whenever the interlaced solution strategy is employed in hybrid simulations,

it is critical that an integration time step is determined that guarantees a smooth displacement command signal generation. Because this time-step size can quickly become very small, it is recommended to instead utilize one of the adaptive strategies discussed next.

5.4.5 Adaptive Simulation Time-Step Strategies

Based on the event-driven control approaches developed by Sandee et al. (2005), some new promising solution strategies for hybrid simulation are proposed below. Furthermore, problems and difficulties encountered with one of the suggested ideas are explained as well. The proposed event-driven solution methods can be divided into two categories: (1) methods that handle asynchronous actions by adapting the simulation time step while keeping the integration time step constant (discussed in this section) and (2) methods that deal with asynchronous actions by adjusting the integration time steps (discussed in the next section).

Skipping Ahead:

One of the problems with the current event-driven solution schemes is that if the computational driver takes a long time to compute the new trial response quantities or the transmission of those quantities is delayed through the network, there remains very little time to correct the actuator positions to the target values. This is because the event-driven controller will generate predicted command displacements up to 80% of the simulation time step, leaving only 20% of such time step for correction. So if the prediction is not extremely accurate, significant corrections need to be applied in very little time. This makes it difficult to generate smooth actuator command signals.

In order to improve the current strategy, one of the main ideas is to use some approximate predictor to not only predict the response a single time step ahead ($+\Delta t_{\text{int}}$), but also generate predictions that extend further out ($+2\Delta t_{\text{int}}$, $+3\Delta t_{\text{int}}$, …). These predicted response quantities correspond directly to the trial response quantities computed in the integration methods. Thus, multiplying the integration time step by the appropriate factors and recalculating the trial response quantities can generate predictions at multiples of the integration time step.

The first solution strategy that is trying to incorporate those additional trial response quantities is shown in Fig. 5.19 above. It is assumed that $Event_{i+1}$ (that has trial displacements extending out to $5\Delta t_{int}$ attached) arrives very close to the end of a simulation time step. As was

previously explained, this means that there would be very little time to correct towards the targets at $t_{int,i+1}$. So, instead of trying to correct towards those displacement targets, the idea is to simply skip this step and head towards the next predicted targets at $t_{int,i+2}$. However, the key question is where the resisting forces required by the integration method should be measured. Since the integrator does not know that the predictor-corrector algorithm is already heading for the trial displacements at $t_{int,i+2}$, the next received resisting forces are assumed to be the forces at $t_{int,i+1}$ (in accordance with all the forces calculated by the analytical elements). Therefore, the resisting forces need to be measured at the target displacements at $t_{int,i+1}$.

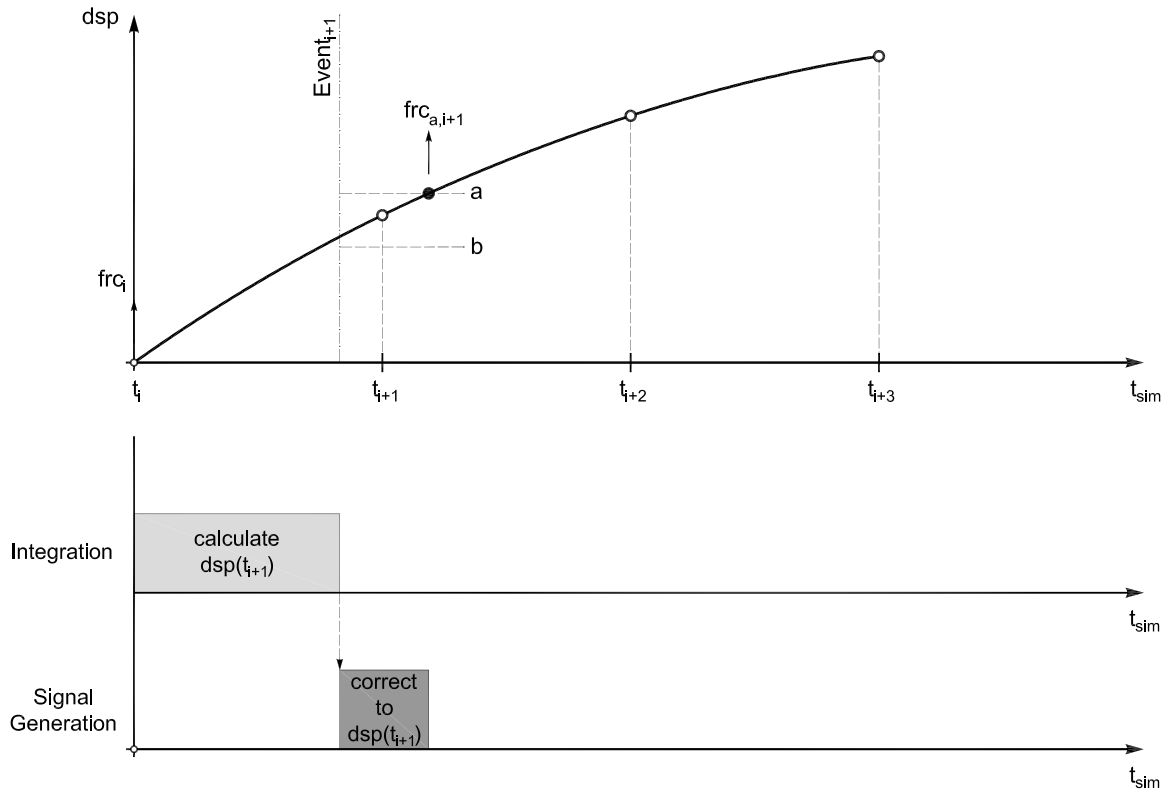


Fig. 5.19 Difficulties with solution strategy that is skipping head.

From Fig. 5.19, it can be seen that for case (a) where the new trial displacement that becomes available with $Event_{i+1}$ is above the current predicted value, the force could simply be measured once that target is reached. On the other hand, in case (b) where the new trial displacement is lower than the current displacement, it is not possible to measure the resisting force anymore, since the predicted displacement shot over the target and is already heading towards the next predicted value. Since there is no guarantee that the new trial displacements are

always above the predicted displacements the event-driven strategy that is based on skipping ahead has to be dismissed and other solution strategies need to be sought.

Constant Correction-Time:

An alternative approach that makes use of the predicted response quantities at multiples of the time step (and that does not possess force measuring problems) uses a constant correction-time. This means that no matter how late an event with the attached trial displacements is received, the correction towards the new targets is executed over a predetermined constant time interval T_C. This time interval is chosen at the beginning of a hybrid simulation. The chosen value should be large enough to allocate enough time for the corrector algorithm to achieve good performance and can be determined from simulations previous to a test.

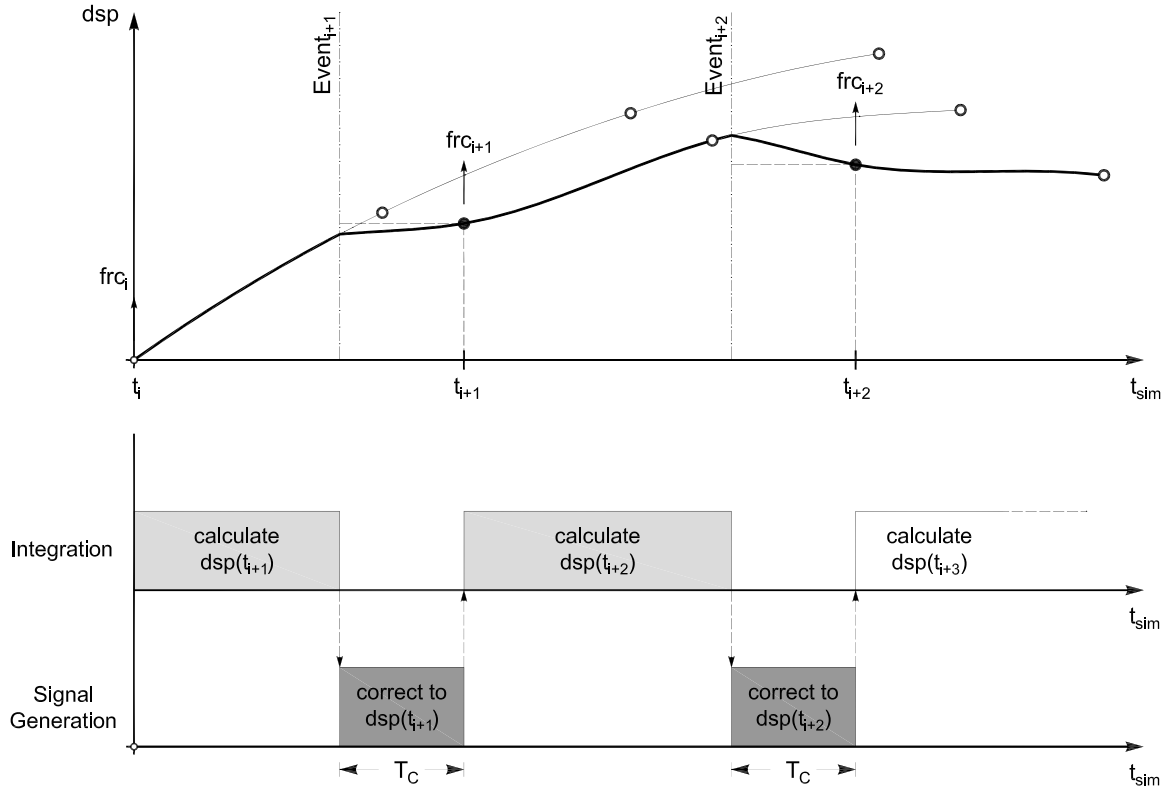


Fig. 5.20 Displacement path and execution sequence of constant correction-time strategy.

From Fig. 5.20 above, it can be seen that once the new trial displacements are obtained at the occurrence of $Event_{i+1}$ the predictor-corrector algorithm is correcting towards the target over the time interval T_C. At the target displacement, the corresponding resisting forces are measured and the predictor starts shooting towards the next available trial value. However, the simulation time step Δt_{sim} is adjusted by taking the length of the previous simulation time steps into

account. The new simulation time step can be computed from the average of all the previous simulation time steps or from a weighted moving average of the previous simulation time steps with gradually increasing weights on the more current time steps.

$$\Delta t_{sim,i+1} = \frac{2}{n(n+1)} \sum_{k=0}^{n-1} (n-k)\, \Delta t_{sim,i-k} \tag{5.28}$$

where n is the total number of previous simulation time steps utilized for the calculation of the next simulation time step. The purpose of adapting the simulation time step in such a manner is to reduce the possibility that an event (like $Event_{i+2}$ in Figure 5.20) is received passed the next predicted trial displacement. Since this might cause a load reversal and should therefore be avoided, some additional safety factor could be used to increase the average simulation time step computed in Equation (5.28). Furthermore, using this approach the simulation time step will adapt to the variable network communication speed. This event-driven solution strategy should thus be employed only if the experimental portions of a structure exhibit truly velocity-independent behaviors.

5.4.6 Adaptive Integration Time-Step Strategies

The adaptive simulation time-step strategy introduced in the previous section improves several aspects of traditional solution schemes. However, some of the problems remain still unresolved. One of these problems is that by adapting the simulation time step, the hybrid simulation does not run at a uniform speed but continuously slows down and speeds up depending on the occurrence of the events. This produces velocities in the experimental subassemblies of a structure that are no longer related to the real velocities (meaning that they are equal to or a constant portion of the real velocities). Thus, two additional solution strategies that are based on adapting the integration time steps (and therefore resolve this problem) are introduced and discussed in this section. Both of these methods rely again on multiple predictions that extend out several time steps ahead.

Adaptation by Multiples of Integration Time Step:

The first scheme that is proposed here adapts the integration time steps by multiples of the initial integration time step. This means that if no event is received by the time the first predicted displacement value is reached, the algorithm just keeps predicting towards the next values at

$t_{int,i+2}$, $t_{int,i+3}$ and so on. Once the new sequence of trial displacements is received from the integration method, the predictor-corrector algorithm starts correcting towards the corresponding new target. So, if the $Event_{i+1}$ is received after Δt_{sim} but before $2\Delta t_{sim}$ (as shown in Fig. 5.21), the second displacement of the just-received sequence has to be corrected towards. As soon as the corrector reaches such displacement, the forces are measured. In addition to those measured forces, an integer $\lambda = 2$ is sent in order to inform the integration scheme that the returned forces do not correspond to the trial displacements at $t_{int,i+1}$ but rather $t_{int,i+2}$. With this information available, the integrator then sends the trial response quantities at $t_{int,i+2}$ off to the analytical elements and then assembles the global resisting force vector at such time step from the analytical and experimental element resisting forces. Hence, the integration method is using variable time steps that are multiples of the initially specified integration time step.

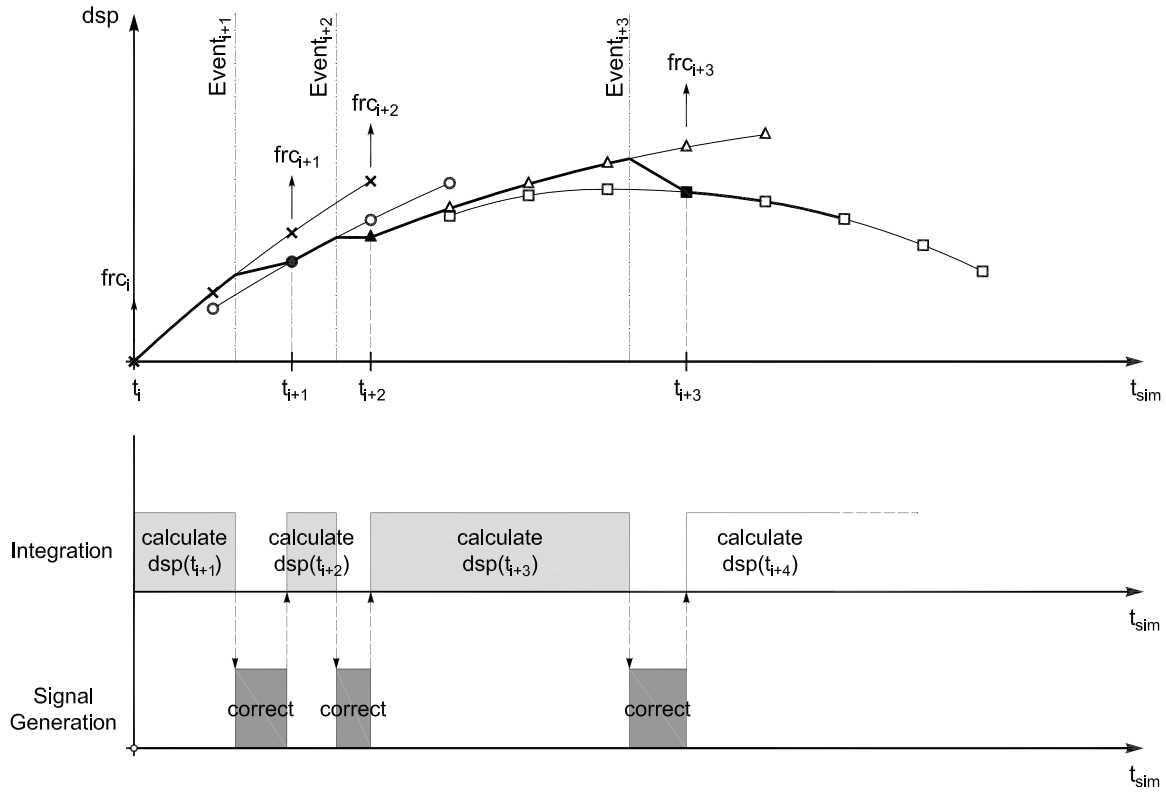


Fig. 5.21 Displacement path and execution sequence of first adaptive integration time-step strategy.

The advantage of this solution strategy is that only small corrections have to be made from the predicted response. Furthermore, since the simulation time step is not changed and no

slow-down or hold states are needed, the hybrid simulation runs at a consistent speed and does not continuously slow down or speed up. The rate at which the simulation is executed is determined by the ratio of the simulation time step over the integration time step. Such time-step values are chosen prior to starting a hybrid simulation and can be determined from previous simulations.

The disadvantages of this strategy are that the machine running the computational driver is not working to full capacity and the trial response quantities cannot be sent to the analytical elements before the resisting forces are received from the experimental subassembly. Furthermore, it is not yet clear how the integration time steps could be adapted in a hybrid simulation with multiple experimental subassemblies. Difficulties arise when the forces sent back from different subassemblies no longer correspond to the same time step. This occurs when the experimental subassemblies did not receive the events at the same time and the predictions, consequently, do not extend out by equal number of steps.

Adaptation Depending on Execution Time:

An alternative approach to the previous solution strategy is to adjust the integration time steps not by multiples of an initial time step but by a variable length that depends on the previously consumed computation plus transmission time.

With reference to Fig. 5.22, it can be seen that if $Event_{i+1}$ is not received by the time the displacement commands reach the first of the predicted trial values, the forces and the displacements are measured anyways. The prediction, however, is continued towards the trial displacement at $t_{int,i+2}$. Once the new trial displacements are received from the computational driver, the previously measured forces and displacements as well as the time consumed by the response analysis task, $T_{RAT,i+1}$, are sent back to the integration method. Since the forces were measured at the wrong displacements, the error Δ_{dsp} between the measured and the calculated displacements are used to correct the measured resisting forces utilizing the initial stiffness matrix (or if available the tangent stiffness matrix) of the experimental subassembly. This correction is similar to the initial stiffness modification that was proposed by Nakashima and Kato (1987) to mitigate experimental errors generated by control systems. Once the integration method has solved the equations of motion for the current time step, $\Delta t_{int,i+2} = T_{RAT,i+1}$ is used to create the new trial displacements at multiple future time steps.

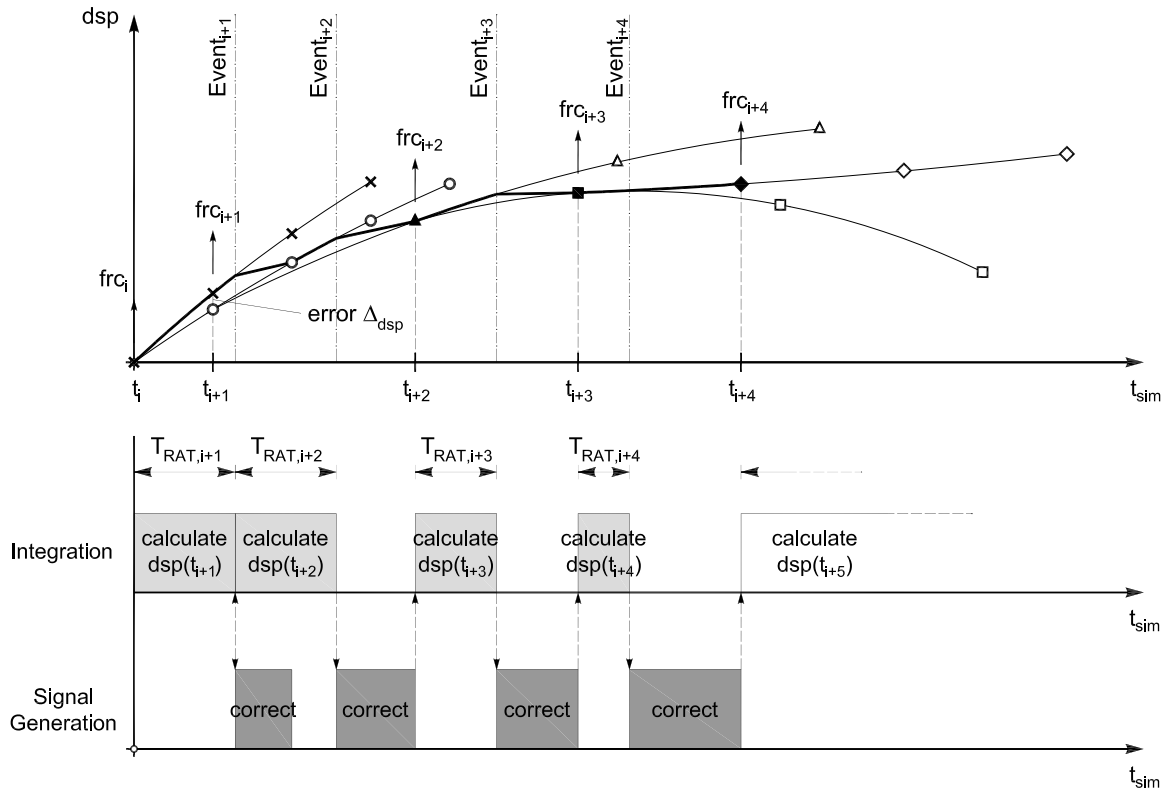


Fig. 5.22 Displacement path and execution sequence of second adaptive integration time-step strategy.

The advantages of this method are, again, that only small corrections have to be made from a predicted response and that the ratio of the simulation time step to the integration time step is not changed. Furthermore, the trial response quantities can now be sent to the analytical elements at the same time the predicted trial response quantities are sent to the experimental subassemblies. This means that the analytical elements can perform their computations at the same time that the control systems are imposing the displacement commands on the experimental subassemblies. Additionally, this solution strategy is still applicable in a hybrid simulation with multiple experimental subassemblies. In such tests, the new integration time steps can be adjusted by considering all the response analysis task times T_{RAT} from all the different subassemblies. Either the maximum or the average of the response analysis task times can be used in the determination of the new integration time step. Thus, the two main problems of the previously proposed solution strategy have been solved. Moreover, the third disadvantage of the previous scheme, which was the rather extensive idle times of the computational driver, has been improved (as can be seen from the execution sequence in Fig. 5.22 above).

5.5 SUMMARY

The predictor-corrector algorithms and the event-driven solution strategies covered in this chapter provide the required synchronization between the integration and actuator control processes, which intrinsically run at different time rates. After a brief review of previous developments a wide range of new predictor-corrector algorithms were discussed. These algorithms achieve improved performance in the sense that they generate continuous displacement command signals without the inconsistencies in the switching interval. Furthermore, two predictor-corrector algorithms that are based on velocities and accelerations in addition to displacements have been introduced. It has been shown that they achieve improved accuracy while maintaining the continuity of the generated displacement command signals. The theory of such algorithms was covered in detail and their performance and accuracy were evaluated graphically. All of these synchronization predictor-corrector algorithms have been implemented in the C programming language and are available to users for performing continuous hybrid simulations.

The event-driven solution strategies, which utilize the previously described algorithms for prediction and correction, were investigated next. Again the existing strategy was evaluated first, and then based on the determined deficiencies new and improved solution strategies were developed. These strategies are based on adaptive control concepts where parameters of the event-driven controller, such as time steps, actuator velocities, etc., are adjusted during simulation according to some suitable performance criteria. One of such solution strategies was specifically developed to work in conjunction with the implicit Newmark direct integration method (see Chapter 4), by adapting time intervals over which displacement commands are imposed. Except for the adaptive strategies that have not been released yet, all the other event-driven strategies have been implemented in Simulink/Stateflow (Mathworks) and are available to users for performing hybrid simulations.

6 Hybrid Simulation Case Studies

6.1 INTRODUCTION

Case studies of two novel applications of hybrid simulation are presented in this chapter. The main purpose is to demonstrate and validate the features and the flexibility of the redesigned and newly implemented object-oriented hybrid simulation framework. The two case studies were performed at the μNEES laboratory at the University of California, Berkeley, and utilize a wide range of methods and implementation details for hybrid simulation.

6.2 RAPID GEOGRAPHICALLY DISTRIBUTED TESTS

The latest advancements in networking technology paired with the experimental software framework OpenFresco make fast geographically distributed hybrid simulations finally feasible. As part of the George E. Brown, Jr. Network for Earthquake Engineering Simulation (NEES) program, a national collaboratory, called the NEESgrid, has been developed. NEESgrid is based on the Internet2's Abilene network and links 15 NEES equipment sites together. The Abilene network is a private high-speed backbone network, which is used for education and research and has (as of 2006) a bandwidth of up to 10 Gbit/sec. For the fast geographically distributed hybrid simulations which were performed between NEESinc in Davis, California, and the μNEES laboratory of the *nees@berkeley* equipment site, one of the 1 Gbit/sec connections of the NEESgrid network was utilized. The subsequently described tests demonstrate how OpenFresco (see Chapter 3) in combination with a high-performance FE-software can be used to execute soft real-time hybrid simulations across large network distances.

6.2.1 Motivation

Over the last decade the introduction of high-performance devices, to improve structural behavior of components and/or systems under extreme loading conditions such as earthquakes, blast, and wind, has increased notably. Most of these devices, such as base-isolation bearings,

energy-dissipation devices, and high-performance materials, exhibit intrinsic rate-dependent behavior. Thus, it is of utter importance that testing of structural systems incorporating such devices is performed at the correct loading rates. This means that full-scale hybrid simulations need to be executed in real-time. Furthermore, if the hybrid model happens to be a scaled version of the prototype model with a length scale factor of S_l and additionally velocities need to be preserved, meaning that the velocity scale factor $S_v = 1$, hybrid simulations are required to be executed S_l^{-1} times faster than real time. On the other hand, if strain rates instead of velocities are to be preserved for the scaled model structure, experimental tests need again to be executed in real time, since the time scale factor $S_t = 1$ is independent of the length scale S_l.

Table 6.1 Similitude laws for rapid testing.

Physical Quantity	*Dimension*	*Scaling Factor with S_l, $S_E = S_v = 1$*	*Scaling Factor with S_l, $S_E = S_{\dot{\varepsilon}} = 1$*
Length, l	L	S_l	S_l
Displacement, d	L	S_l	S_l
Velocity, v	LT^{-1}	1	S_l
Acceleration, a	LT^{-2}	S_l^{-1}	S_l
Force, F	F	S_l^2	S_l^2
Time, t	T	S_l	1
Modulus, E	FL^{-2}	1	1
Stress, σ	FL^{-2}	1	1
Strain, ε	1	1	1
Strain-Rate, $\dot{\varepsilon}$	T^{-1}	S_l^{-1}	1
Mass, m	$FL^{-1}T^2$	S_l^3	S_l
Damping, c	$FL^{-1}T$	S_l^2	S_l
Stiffness, k	FL^{-1}	S_l	S_l
Period, T	T	S_l	1
Frequency, f	T^{-1}	S_l^{-1}	1

These similitude laws for the two scaling factor triplets (S_l, S_E, S_v) and (S_l, S_E, $S_{\dot{\varepsilon}}$) are shown in Table 6.1. A different important class of applications for which it is crucial to have the ability to execute a test in real time are hybrid simulations that involve shaking tables. This

means that the physical portion of the hybrid model is tested on a shaking table, which receives real-time displacement commands from the numerical portion of the structure and feeds back the corresponding resisting forces to such (see Chapter 2).

6.2.2 Test Structure and Earthquake Record

Because of its simplicity and nominal computational effort, a one-bay-frame model with one ductile and one elastic column connected by a fairly soft spring was selected for these hybrid simulations. As can be seen from Fig. 6.1, the one-bay-frame had a height of 50 inches, which corresponds to the height at which the μNEES actuators are attached to the experimental steel specimen. The width of the frame was arbitrarily chosen to be 100 inches.

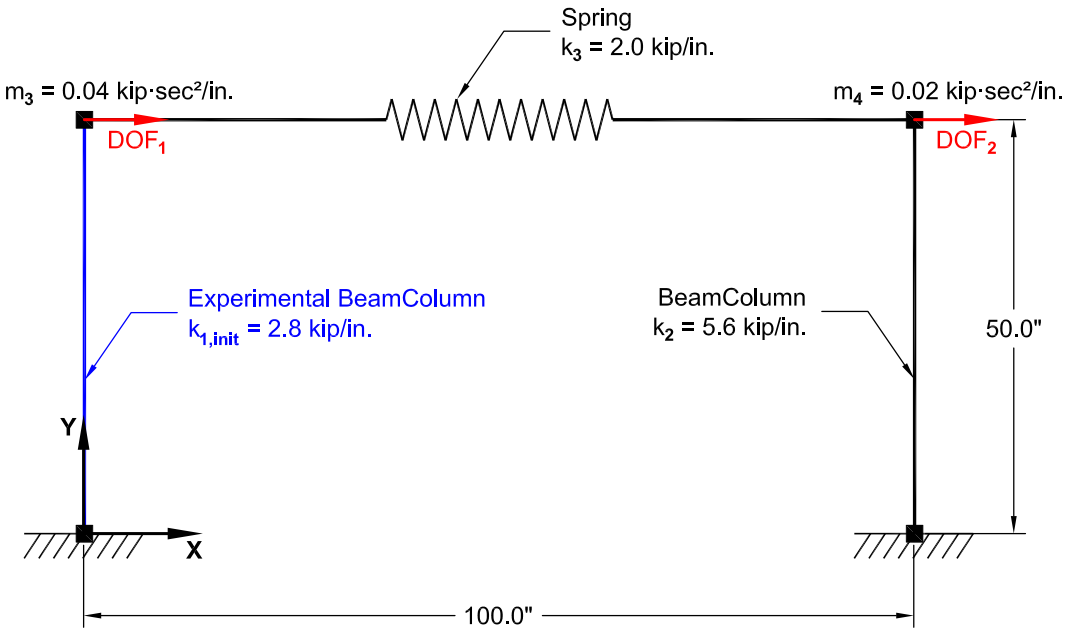


Fig. 6.1 One-bay-frame model with structural properties.

For the hybrid simulation, the left-hand column was represented by a specimen in the μNEES laboratory, and the right-hand column and horizontal spring connecting the two columns were modeled numerically. Furthermore, the left experimental column, the right linear-elastic column, and the linear-elastic spring connecting the two columns were assigned stiffnesses of $k_{1,init}$ = 2.8 kip/in., k_2 = 5.6 kip/in., and k_3 = 2.0 kip/in., respectively. The mass on top of the left column was 0.04 kip·sec^2/in. and the mass on top of the right column was 0.02 kip·sec^2/in. Given that only the horizontal degrees of freedom were considered, this simple one-bay-frame model had a total of two free degrees of freedom, both with mass. With the selected structural

properties, the resulting vibration periods of the model were T_1 = 0.622 sec and T_2 = 0.315 sec for the first and second mode, respectively. To get significant contributions to the total response from the second mode, in addition to that from the first mode, the linear-elastic spring was given a fairly low stiffness. In addition, the damping matrix was chosen to be proportional to the mass matrix instead of the stiffness matrix, so that the second mode would be less damped than the first. The first mode was assigned a damping value of 5% of critical, which led to a 2.53% damping ratio in the second mode.

The north-south component of the El Centro ground acceleration history recorded during the 1940 Imperial Valley earthquake was used to dynamically excite the model. The 32 sec long time history consisted of 1600 data points recorded every 0.02 sec and was scaled to a pga of 0.319 g. The elastic response spectra of such motion for damping ratios of 2.5% and 5% and with markers at the two model periods are shown in Fig. 6.2.

6.2.3 Test Setup and Procedure

As can be seen from Fig. 6.3, the experimental setup that was utilized to test the physical column of the one-bay-frame model consisted of a self-equilibrating reaction frame that was designed to support the actuator and the specimen under static and dynamic loading conditions. According to Mosqueda (2003), it was found that the reaction frame had a fundamental natural frequency of 37 Hz, which was considered sufficiently high to minimize interactions with the fairly flexible test specimen. The 50-inch high experimental S4x7.7 steel column specimen was mounted on top of a special clevis with replaceable steel coupons that was capable of simulating the plastic hinge zone at the end of a steel section. Equipped with two ductile ASTM A36-04 (f_y = 69 ksi) steel coupons, this connection produced stable and repeatable hysteresis loops that were ideal for testing and evaluating new ideas and concepts in hybrid simulation. In addition, a wide variety of hysteretic behavior can be simulated by using different coupon designs and/or nut configurations as described in Rodgers and Mahin (2004). The connection details and the hysteresis loops (SAC loading protocol) for the configuration with two ductile steel coupons with top and bottom nuts are shown in Fig. 6.4.

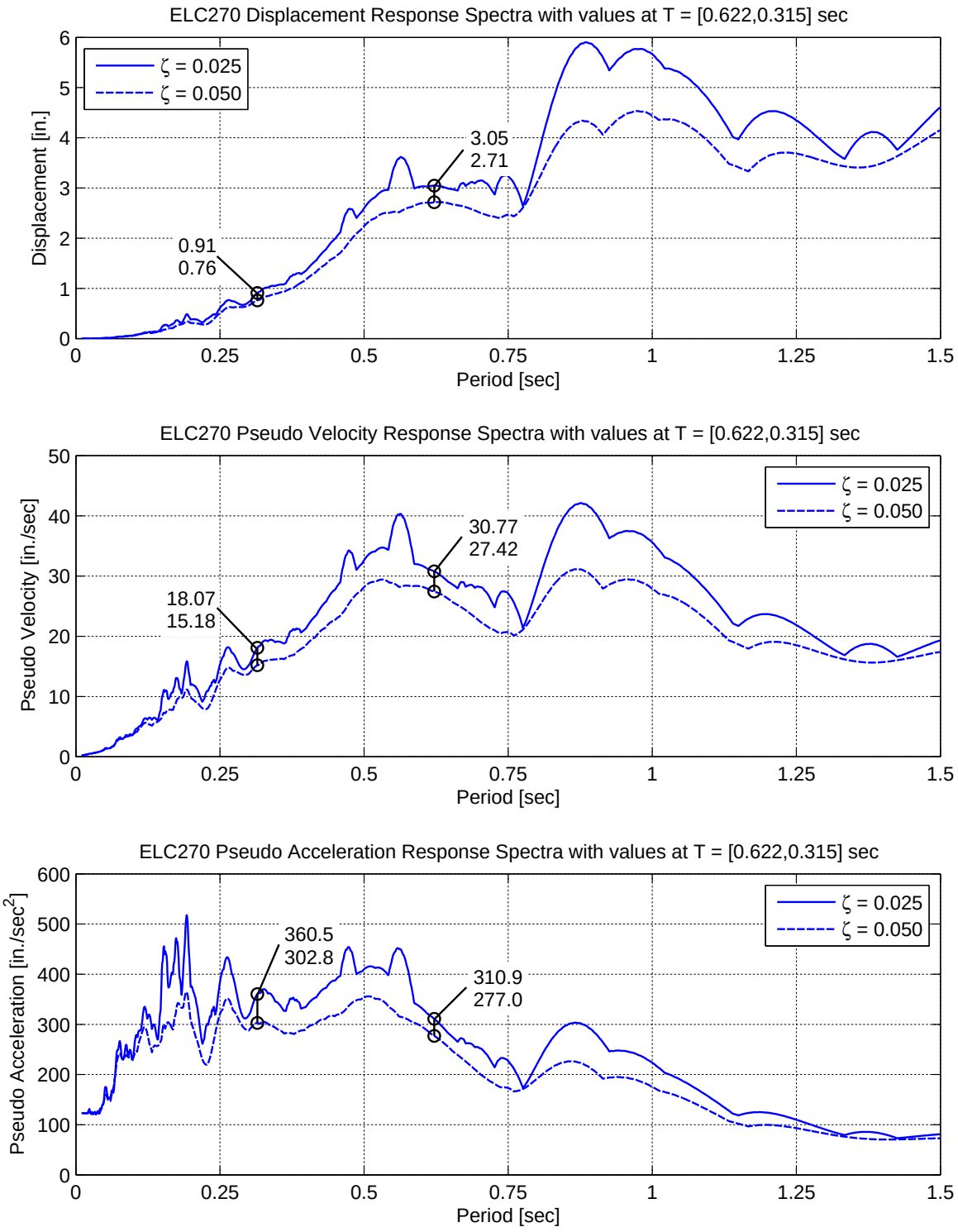
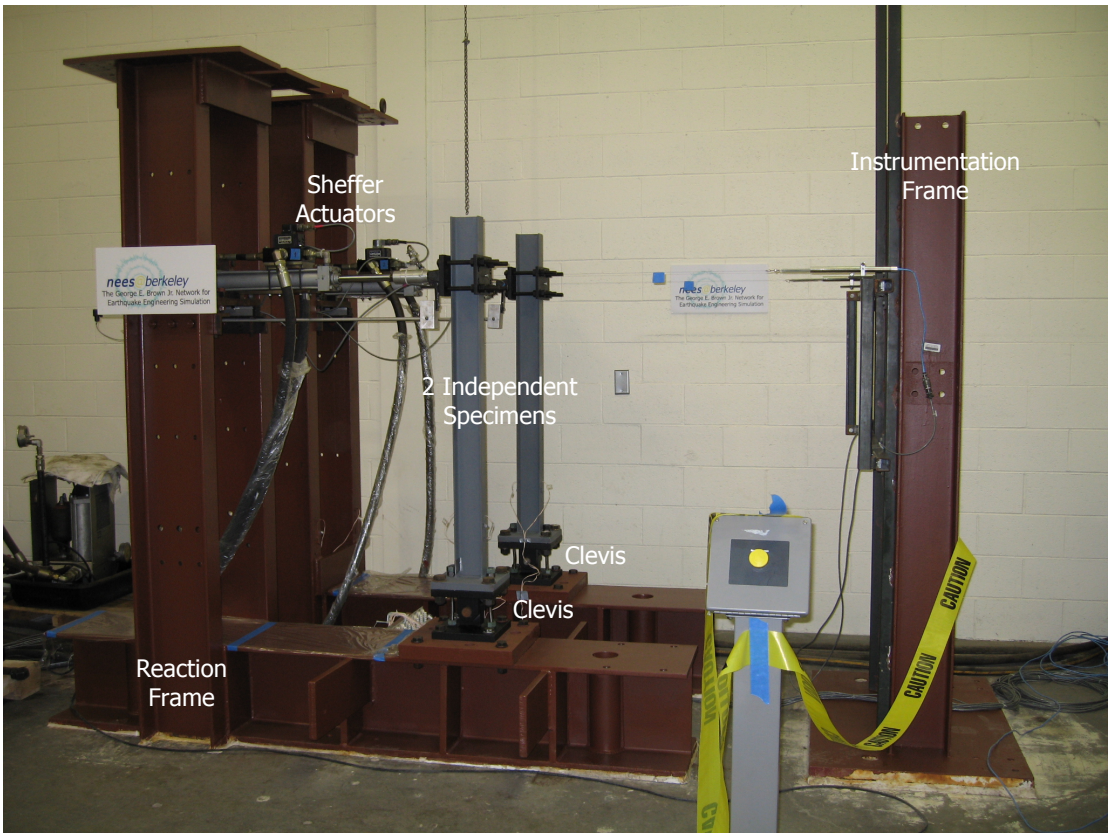


Fig. 6.2 Elastic response spectra for north-south component of 1940 El Centro, Imperial Valley earthquake record, with values at model periods.



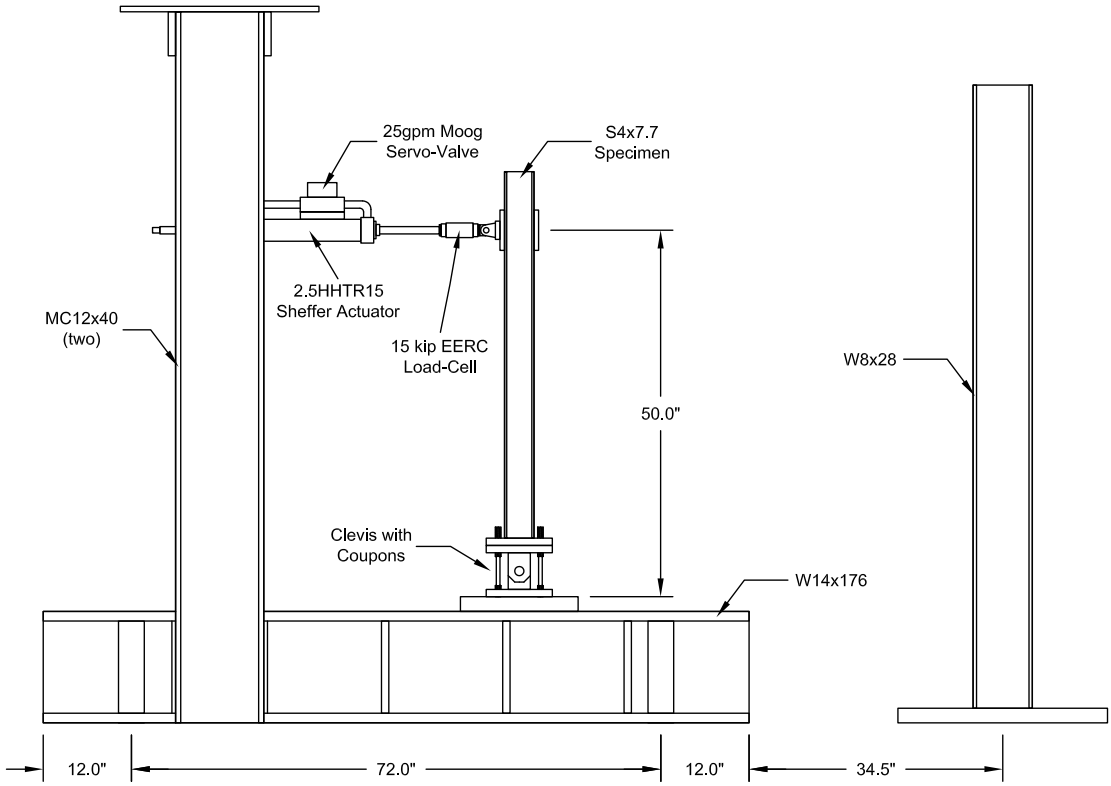


Fig. 6.3 μNEES experimental setup with two independent specimens.

The 12 kip dynamic actuator that was used to load the specimen during the hybrid simulation tests was a double-ended 2.5HHTR15 Sheffer model with a total stroke capacity of 15 inches. The actuator was driven by a 25 gpm Moog servo-valve connected to a 3000 psi hydraulic oil supply line, which yielded a velocity limit of roughly 23 in./sec. Given the bulk modulus of the hydraulic oil, β = 100 ksi, and the approximate weight of the payload, W = 0.049 kip, the oil column frequency of the μNEES setup can be estimated at f_{oc} = 148 Hz.

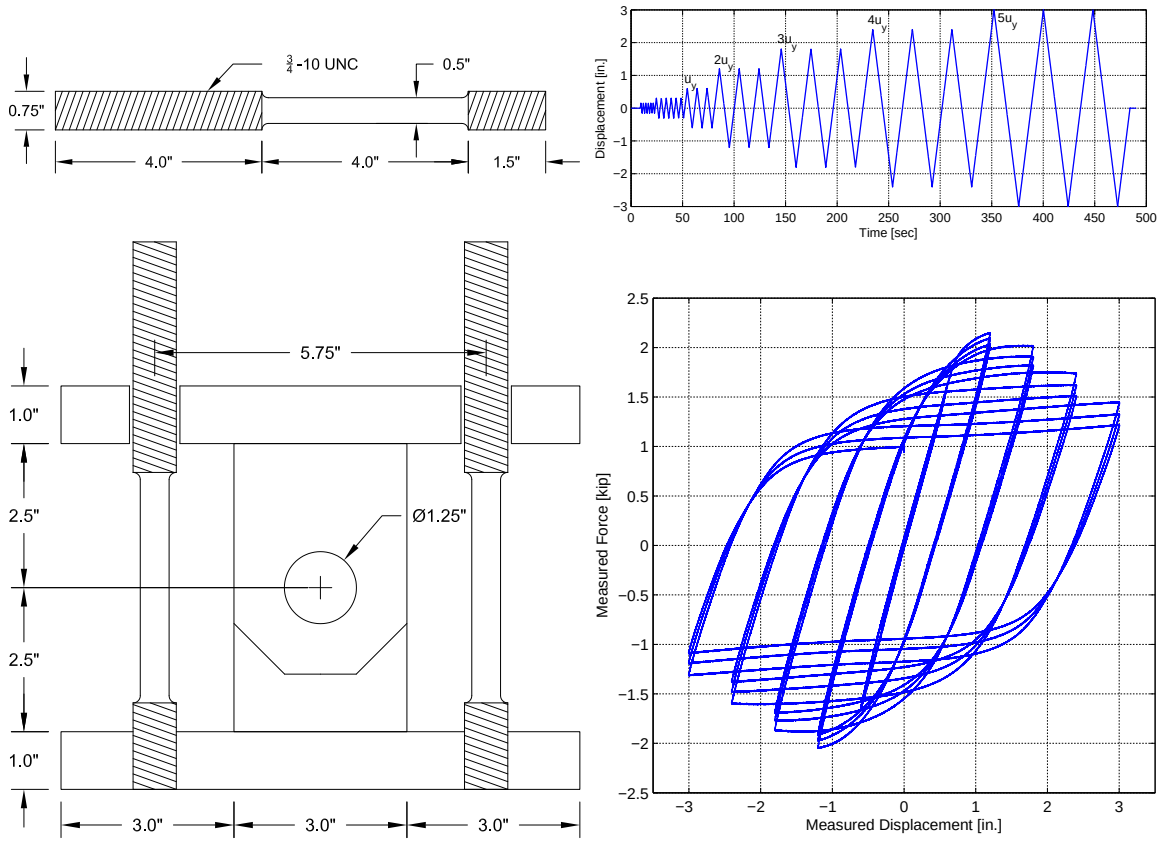


Fig. 6.4 Connection with coupon details and displacement history with hysteresis loops.

A unidirectional aluminum load cell with a capacity of ±15 kip was mounted between the actuator end and the specimen in order to measure the experimental resisting forces. The actuator displacement feedback, required by the servo-hydraulic control system, was provided by a Novotechnik TLH500 displacement transducer that had a displacement limit of ±10 inches. Selected properties of the μNEES setup are summarized in Table 6.2.

Table 6.2 Properties of μNEES experimental setup.

Hydraulic properties	*Values*
Supply pressure	p_s = 3 ksi
Return pressure	p_r = 0.05 ksi
Bulk modulus of oil	β = 100 ksi
Sheffer 2.5HHTR15 actuator properties	*Values*
Actuator bore	b = 2.5 in.
Rod diameter	d = 1.0 in.
Rod length	L = 40 in.
Actuator stroke	S = 15 in.
Servo-valve & payload properties	*Values*
Flow rate	q = 96.25 in^3/sec
Weight (rod, load cell, specimen)	W = 0.049 kip
Derived experimental setup properties	*Values*
Displacement limit	u_{max} = ±7.5 in.
Velocity limit	v_{max} = ±23.3 in./sec
Oil column stiffness	K_{oc} = 110 kip/in.
Oil column frequency	f_{oc} = 148 Hz
Reaction frame frequency	f_{rf} = 37 Hz
Specimen frequency	f_{sp} = 25.6 Hz

For the operation of the described experimental setup a MTS 493 real-time controller with the Structural-Test-System (STS) software and the xPC Target real-time environment was utilized. This digital controller provided closed-loop PID, differential Feedforward, and Delta-P control capabilities. The accompanying STS software is a graphical user interface that allows the operator to access the controller for calibration, configuration, test setup, and data display. In order to achieve the best possible performance of the servo-hydraulic control system throughout the real-time geographically distributed hybrid simulations, some fair amount of time was spent prior to the tests, tuning the actuators locally. The established PID and feed-forward gains, shown in Table 6.3, delivered the fastest possible system response with minimal actuator lag and overshoot, and without excessive excitation of the actuator oil-column frequency. Any attempt to raise the gains past the given values initiated the generation of high-frequency actuator vibrations

at the oil-column frequency, which in turn created significant force oscillations that fed back into the hybrid simulation.

Table 6.3 Parameters of real-time MTS 493 controller.

Controller parameters	*Values*
Proportional gain	p = 12.00
Integral gain	i = 0.00
Integrator authority	i_A = 5%
Derivative gain	d = 0.07
Feedforward gain	f = 0.13
Bandwidth of low pass filter	bw = 1024 Hz
Delta-P gain	Δp = 0.00

The soft real-time geographically distributed hybrid simulation described here was executed between the NEESinc Headquarters in Davis, California, and the *nees@berkeley* equipment site in Richmond, California. The OpenSees finite element software was used to model and analyze the one-bay frame structure on a portable computer at the NEESinc Headquarters, the μNEES setup was utilized to test the experimental portion of the model at the *nees@berkeley* laboratory, and the experimental software framework OpenFresco was employed to connect the two across the 60 miles long network connection. The specific client-server configuration for this test, described in detail in Chapter 3, can be seen from Fig. 6.5. It should be noted that the ShadowExpSite and ActorExpSite pair of OpenFresco provided the necessary capabilities to distribute the two processes across a network using different communication protocols.

The OpenSees finite element analysis software in combination with OpenFresco's tightly integrated experimental element and experimental site objects composed the client side of the distributed hybrid simulation. The explicit Newmark method (see Chapter 4) was chosen for the integration of the equations of motion in OpenSees because of the following reasons. First of all, since both free degrees of freedom of this one-bay frame model had mass assigned and additionally the structure was chosen to be fairly flexible, there were no modes present with excessively high frequencies. This allowed for the selection of an explicit conditionally stable integration method, with a reasonable integration time-step size. Secondly, given that the goal of

this case study was to perform a geographically distributed hybrid simulation in real time, it was desired to choose an integration method that did not require iteration and needed as little time for computation as possible. As such, the explicit Newmark method was the ideal choice for the problem at hand. Moreover, it was possible to use the unmodified equations of motions (without inertial force compensation) for slow hybrid simulations (see Chapter 2) because the actual physical mass of the specimen was only 0.3% of the analytical mass at node 3. The test results presented in Section 6.2.5, successfully validated this assumption. The TwoNodeLink experimental element available from the OpenFresco element library was used to represent the physical column tested in the laboratory.

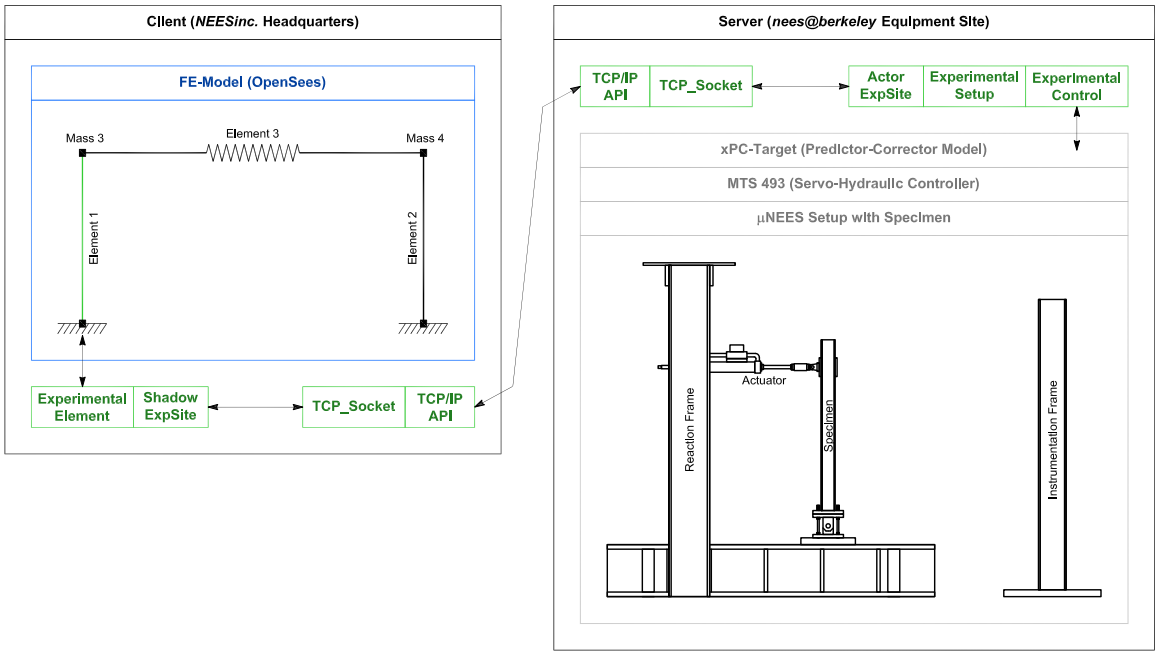


Fig. 6.5 Client-server configuration.

The server side of the hybrid simulation consisted of OpenFresco's experimental site, experimental setup, and experimental control objects. It should be noticed that for this specific validation test the finite element analyst's view of a hybrid simulation was chosen. This means that the OneActuator experimental setup was located on the server side of the distributed configuration and hence all the response quantities transmitted across the network were in the element's basic reference coordinate system. Lastly, the xPCtarget experimental control object was employed to connect to the real-time xPC Target machine, which was running the event-driven predictor-corrector model. As described in detail in Chapter 5, the predictor-corrector

model was responsible for handling possible delays caused by the nondeterministic TCP/IP network transactions and the direct integration of the equations of motion.

The first step of the test procedure was to start the OpenFresco lab-server, which in turn initialized and started the predictor-corrector model on the real-time xPC Target machine. During this initialization, the predictor-corrector model was automatically stopped allowing for last minute adjustments of the servo-hydraulic control system to zero out displacements and forces. This guaranteed that the initial conditions of the experimental element were as close to zero as possible. Once the desired initial conditions were set, the MTS 493 controller was started to accept command signals over SCRAMNet+, which concluded the test initialization on the server side in the *nees@berkeley* laboratory. All that remained to be done to run the rapid geographically distributed hybrid simulation was to start the FEA software (OpenSees) on the client side at the NEESinc Headquarters.

6.2.4 Network Performance Evaluation

To establish a connection between the client and server program of a distributed hybrid simulation, OpenFresco's ShadowExpSite and ActorExpSite classes employ a channel object. The channel object is an abstract base class that defines endpoints of communication from which data leave or enter a program. Consequently, it is the derived subclasses that implement the different communication protocols (sockets) that can be used to transmit data between a client and server program. The client-server architecture with all the available options was described in detail in Chapter 3. For the rapid geographically distributed hybrid simulation case study described here, the TCP_Socket object was utilized to setup a persistent TCP/IP connection from the client application to the server application. The TCP/IP connection guaranteed that the transmitted messages were never lost, damaged, or received out of order. In addition, the persistent connection provided significant performance gains, since the TCP connection setup delay caused by the client server handshake occurred only once during initialization of the distributed test rather than at the beginning of every single transaction.

Besides the use of a persistent TCP/IP connection, the size of the data packages sent to the TCP stack also significantly affected the network performance. Because the TCP protocol has a data stream interface that permits applications to send data of any size to the TCP stack (even a single byte at a time), it was imperative to determine a TCP segment size that would

produce the best possible network performance. There was a trade-off in determining such optimal segment size. If TCP segments were too large, they were likely to become fragmented at the IP level. On the other hand, if they were too small, the network performance degraded greatly because small amounts of data were being sent with at least 40 bytes overhead from the TCP and IP headers. So by increasing the size of the data packages that were transmitted between the ShadowExpSite and the ActorExpSite past the minimal size necessary for the response vectors, the so-called sender silly window syndrome was avoided. Sending a storm of tiny data packages is very inefficient and can even disrupt other network traffic (Gourley et al. 2002).

In order to investigate the effect of the data package size on the network performance a sequence of hybrid simulations, where the experimental column of the one-bay-frame model was simulated, instead of tested in the laboratory, was executed across different network configurations. The distributed hybrid simulation tests were performed on a local area network at the *nees@berkeley* equipment site as well as on the Abilene network between Davis and Berkeley. For each test the total simulation time required for the execution of the 1599-step-long direct integration analysis was recorded and used as a measure to evaluate the performance of the network under investigation.

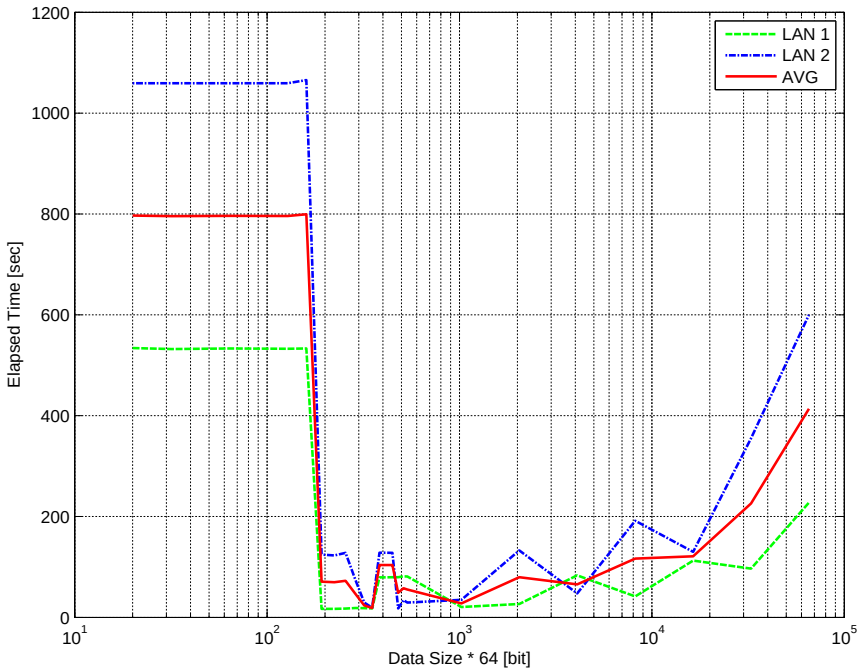


Fig. 6.6 Network performance depending on data package size.

As can be seen from Fig. 6.6, the simulation time decreased abruptly by about a factor of 10, once a data size of approximately 200 64-bit doubles was reached. As expected the performance started deteriorating again for very large data packages, but at a much slower rate than the sudden increase encountered before. As a result of these performance tests, a vector size of 256 64-bit doubles was chosen for the rapid geographically distributed hybrid simulation of the one-bay-frame model. It is important to recognize that the event-driven predictor-corrector model, which executed on the real-time xPC Target machine (see Chapter 5), actually determined how fast the hybrid simulation was running under normal operating conditions. As long as there were no excessive network and integration delays, the simulation time step that was set in the predictor-corrector model, established how much time would pass on a wall clock for each integration time step.

Because the goal in this case study was to perform a hybrid simulation in real time, the simulation time step, Δt_{sim}, had to be set equal to the integration time step, Δt_{int} = 20 msec. However, because the simulation time step had to be a multiple of the controller time step, Δt_{con} = 1/1024 sec, this was not exactly possible. It was decided to choose 20 substeps for each simulation time step. This yielded Δt_{sim} = 19.53125 msec, as long as the network and integration tasks were capable of executing in less than 60% of the predefined simulation time step. Thus, the theoretical testing rate was 0.9766, which turned out to be slightly faster than real time.

The timing and interaction of the different tasks within a simulation time step can be seen from Fig. 6.7. The flow of events was as follows. Once the target displacement was reached at t_i, the resisting force was acquired and sent off across the Abilene network to the integration task. In the meanwhile the signal generation task performed a response prediction that created new displacement commands every Δt_{con}. This guaranteed that the actuator kept moving until the new target displacement was received. After completion of the direct integration of the equations of motion, the new displacement target was sent back from the client across the Abilene network to the server, which triggered the finite-state machine to switch from prediction to correction. As previously mentioned the predictor-corrector model automatically slowed down the actuator movement if the combined network and integration time exceeded 60% of the predefined simulation time step. To achieve optimal performance, this condition needed to arise as little as possible. Once the target displacement was reached at t_{i+1}, the resisting force was measured and the sequence of events started over.

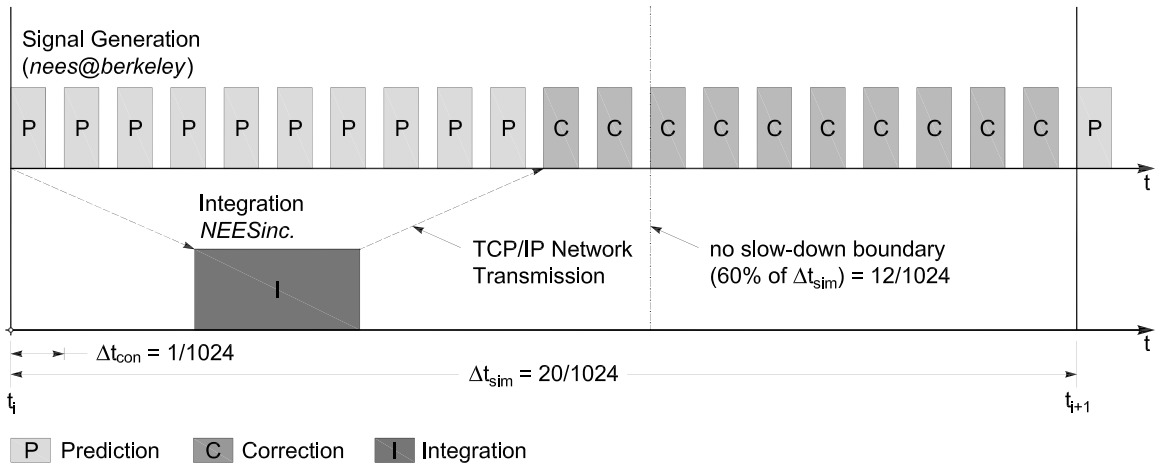


Fig. 6.7 Timing and interaction of tasks within a time step.

During the actual hybrid simulation, timing took place on the client side as well as on the server side of the geographically distributed test. On the client side the time consumed by each step of the transient analysis running on the non-real-time operating system was measured and recorded. On the server side time-stamping of all the predictor-corrector events occurred on the xPC Target machine running in real time. From these results the time required for each simulation time step was later extracted.

As can be seen from Figure 6.8, the distribution of the integration time steps was not as narrow as the one of the simulation time steps. This was because time measurements on the client and server sides were shifted in time with respect to each other. By nature of the predictor-corrector model, simulation time steps on the xPC Target machine could never be shorter than the predefined value; 19.53125 msec in this case. They could, however, be longer for situations where the finite state machine had to slow down the actuator movements due to excessive delays. Contrary to this, the analysis step durations were measured somewhere inside the integration task block in Fig. 6.7, which caused them to vary somewhat more than the simulation time steps.

It can also be observed from Figure 6.8 that the sporadic delays caused by the network and integration tasks occurred at times that were consistent between the two independent measurements. The number of times the finite-state machine had to slow down the actuator due to these delays corresponded to 0.8% of the total number of analysis steps. It should be noted that the time measurements taken on the real-time xPC Target machine are likely more accurate than the ones obtained from the non-real-time client machine. The achieved time-step durations as measures of network performance are summarized in Table 6.4. Because not all the simulation time steps could be completed without any slow downs, the geographically distributed hybrid

simulation was not executed in hard real-time, but rather in soft real time, where sporadic, short delays were encountered. In many applications, such random short delays may be quite acceptable and do not invalidate an entire test.

Table 6.4 Network performance in terms of time-step durations.

	Client-Side	*Server-Side*	*Desired*
dt_{Min} [msec]	16.564	19.531	19.531
dt_{Max} [msec]	107.093	83.984	19.531
dt_{Avg} [msec]	19.877	19.893	19.531
dt_{Std} [msec]	4.156	4.559	0.0
t_{Tot} [sec]	31.784	31.809	31.230

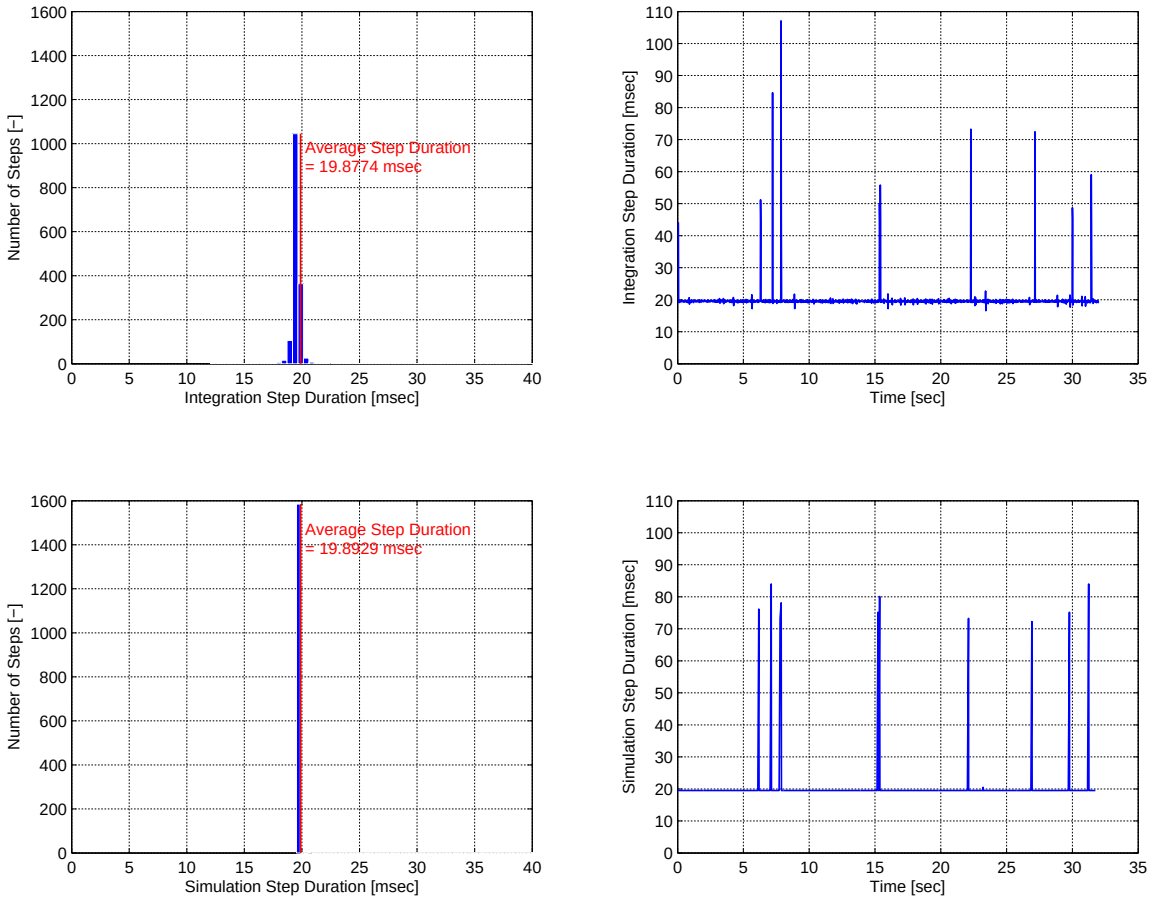


Fig. 6.8 Comparison of integration and simulation step distributions.

6.2.5 Test Results and Interpretation

In this section, the results, as they were recorded by OpenSees on the client side of the hybrid simulation (at the NEESinc Headquarters) are discussed first. Secondly, some predictor-corrector results that were recorded on the xPC Target machine are presented. The measurements recorded by the STS controller software are then used to evaluate the performance of the servo-hydraulic control system that was achieved during the test. Finally, the soft real-time network test is compared to a slow local test to investigate the effect of testing speeds on the response of the one-bay frame structure.

Finite Element Model Results:

Fig. 6.9 shows the lateral displacement, velocity, and acceleration response of the two free degrees of freedom of the one-bay frame structure. The response histories are plotted from the OpenSees output files that were recorded during the real-time test. As marked on the following three graphs, the largest displacement, velocity, and acceleration values seen on top of the left experimental column were 2.31 in., 22.1 in./sec, and 242 in./sec^2, respectively. Because the top of the left experimental column was attached to the actuator of the μNEES setup and the test was executed in real time, the actuator experienced the same response maxima. Consequently, such extremes had to remain inside the actuator displacement limits of ±7.5 in. and the actuator velocity limits of ±23.3 in./sec (as listed in Table 6.2) in order not to saturate the response or trigger the interlocks and shutdown the control system.

Another observation that can be made from the very end of the displacement history is that the structural response kept decreasing with each cycle due to the total energy dissipation, which was determined to be equal to a viscous damping coefficient of approximately 6% of critical. Since at these small displacements the hysteretic energy dissipation completely vanished and because the explicit Newmark method does not introduce any numerical damping, the total energy dissipation was a combination of the assigned viscous damping and the energy dissipation due to experimental errors. Thus, it can be concluded that the additional damping contributed by experimental errors was approximately 1% of critical. This means that the actuator control system was slightly overshooting the reference displacements, which is confirmed from the results acquired by the STS control software.

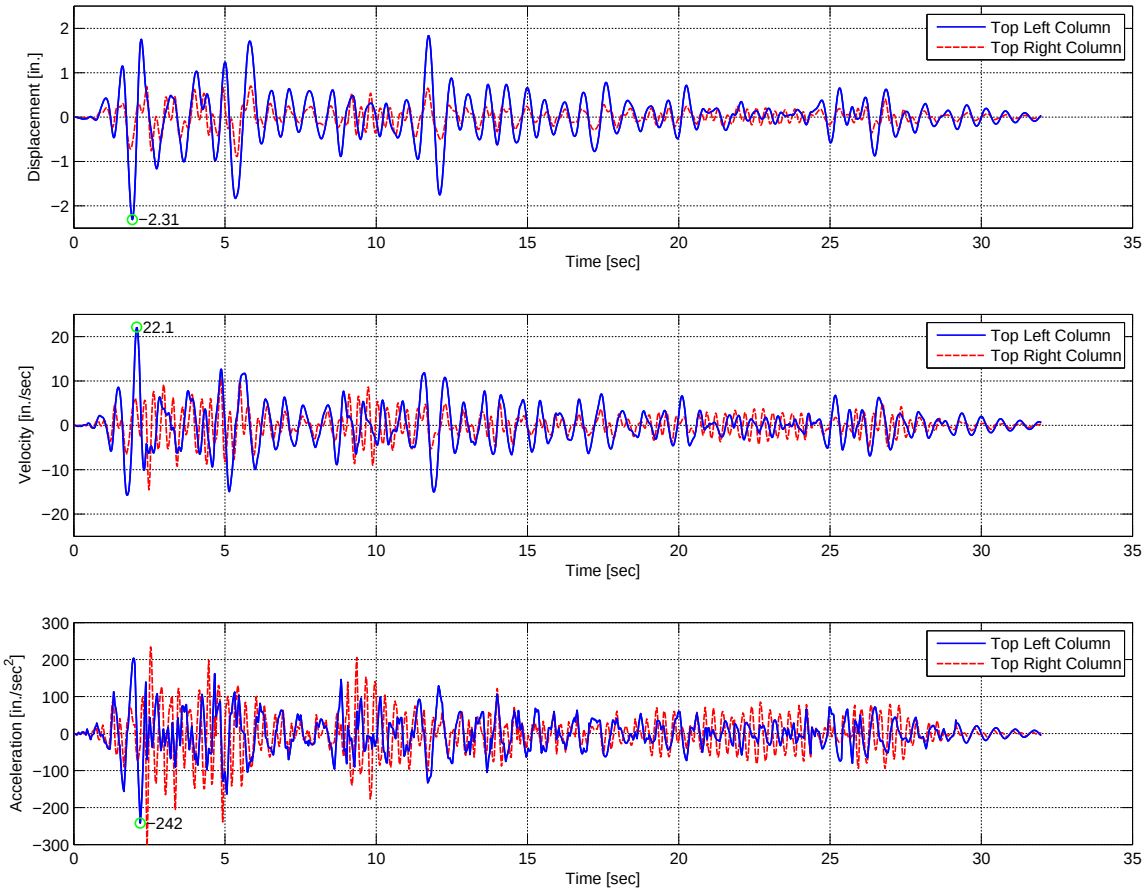


Fig. 6.9 Response histories of one-bay frame model from OpenSees.

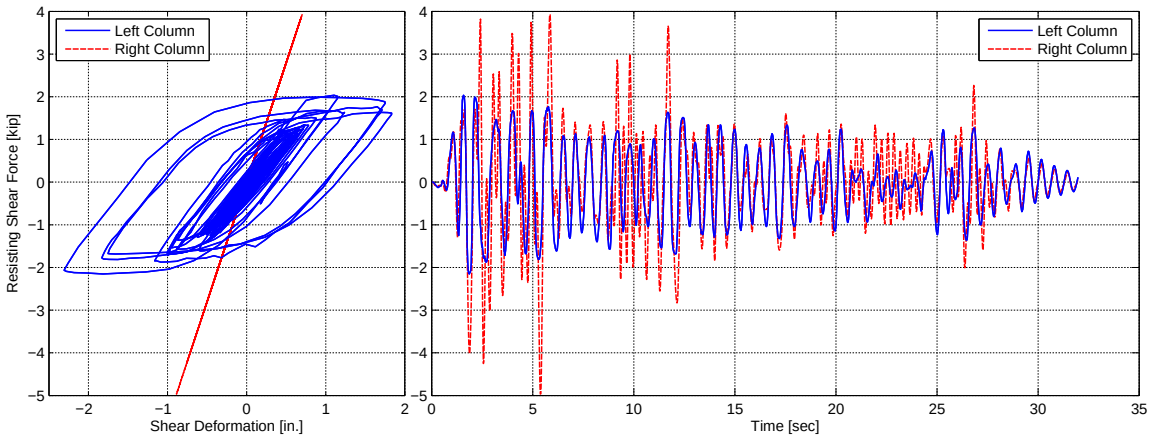


Fig. 6.10 Hysteresis loops and force histories of one-bay frame model from OpenSees.

As can be seen from Fig. 6.10, the resisting forces that were measured by the load cell and fed back to OpenSees are reasonably smooth except at one short instance in time. So even though the hybrid simulation was executed in real time with high velocities and accelerations, force oscillations stayed small. This can be explained by the very small weight (49 lb = 0.3% of

weight at node 3) of the physical components that the actuator had to move forth and back. If the weight had been larger, significant inertial forces would have been generated that in turn would have affected the force measurements to a much larger degree. In addition, it would then have been necessary to compensate for these inertial force contributions in the equations of motion as was explained in Chapter 2.

Predictor-Corrector Results:

To confirm the correct operation of the predictor-corrector finite state machine, Fig. 6.11 provides a closer look at the displacement command generation and the handling of excessive network and computation delays. The smoothness of the actuator displacement command signal (solid red line), which has no jumps and kinks, verifies that the third-order polynomial predictor-corrector model that was chosen for this test was adequate. Furthermore, it can be seen that due to small tracking errors, the measured displacement signal (dash-dotted green line) did not always exactly reach the intersection of the target displacement signal (dashed blue line) with the command displacement signal (solid red line).

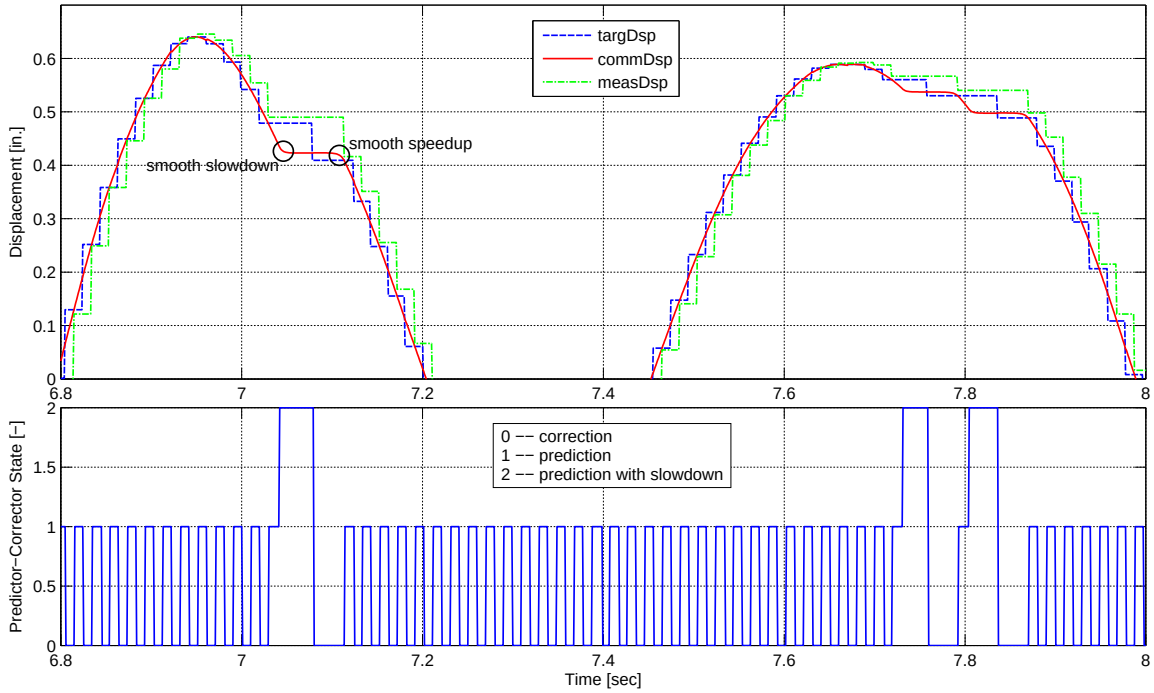


Fig. 6.11 Close up of predictor-corrector state transitions from xPC Target.

Finally, it can be observed that whenever delays occurred, the command displacement signal was smoothly slowed down and then smoothly sped up again when the target

displacement was lastly received. Such smooth transitions of the command displacements are important so as to not generate spurious force oscillation, which would be expected if the actuators were changing their speed abruptly.

Servo-Hydraulic Control System Results:

The performance of the servo-hydraulic control system is evaluated from the displacement and force measurements, which were acquired by the STS control software at a sampling rate of 1024 Hz. As can be seen from Fig. 6.12, the largest displacement error during the rapid hybrid simulation was equal to 0.06 in., which corresponded to approximately 2.6% of the maximal displacement.

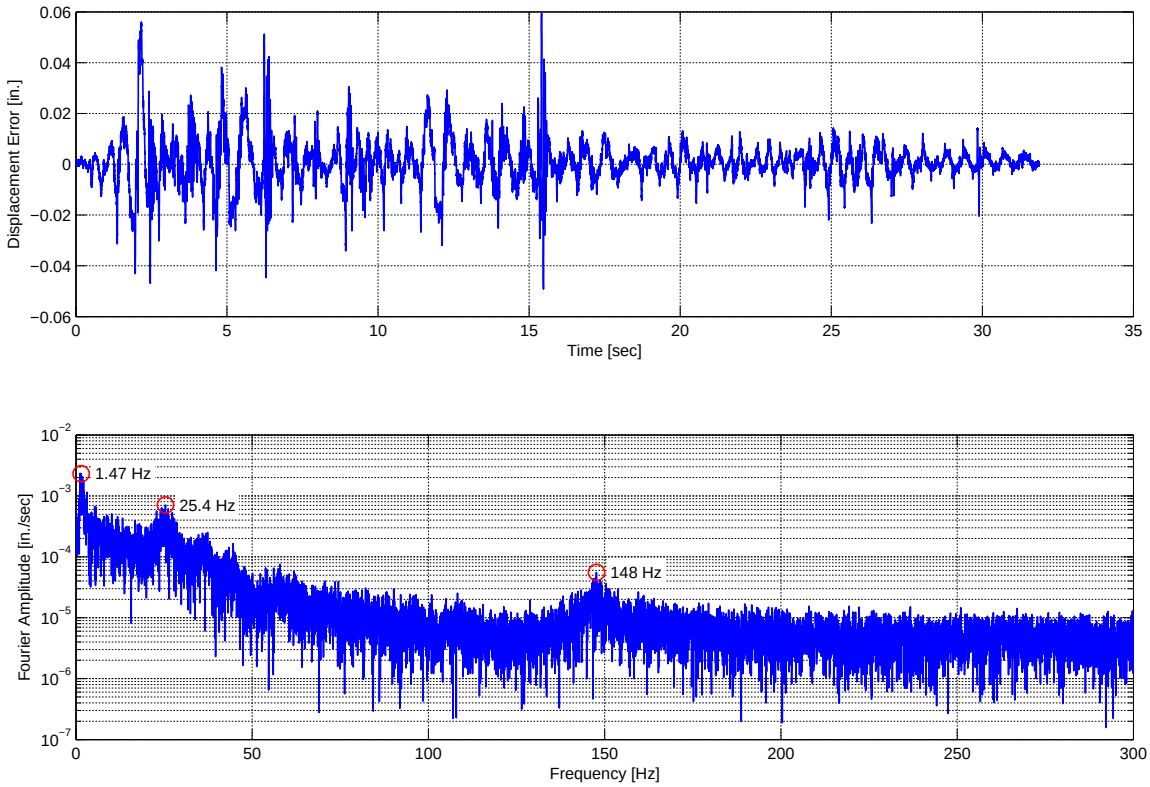


Fig. 6.12 Displacement error history and Fourier amplitude spectrum from STS.

The Fourier amplitude spectrum of the displacement errors clearly shows the fundamental frequency of the structure at 1.47 Hz, which was slightly lower than the linear-elastic frequency of 1.61 Hz, due to the nonlinear softening of the experimental column. The specimen must have generated the second amplitude peak at a frequency of 25.4 Hz, since it was very close to the analytically estimated value of the specimen frequency at 25.6 Hz. Finally, the

third amplitude peak at a frequency of 148 Hz corresponds to the oil column frequency of the μNEES setup and thus confirms the analytically estimated value.

A convenient method to check for actuator lag, lead, undershoot, and overshoot all in one single plot was utilized by Mercan (2007). Fig. 6.13 presents such graph, where the measured displacement is plotted against the command displacement. The following actuator performances can quickly be determined from the graph: If the plot is a straight 45º-line, perfect tracking without any error was achieved. If the plot shows a counter-clockwise or clockwise turning ellipsoid, the actuator produced a lagging or leading displacement response. Finally if the line or the axis of the ellipsoid is turned clockwise or counterclockwise away from the 45º-line, the actuator was undershooting or overshooting.

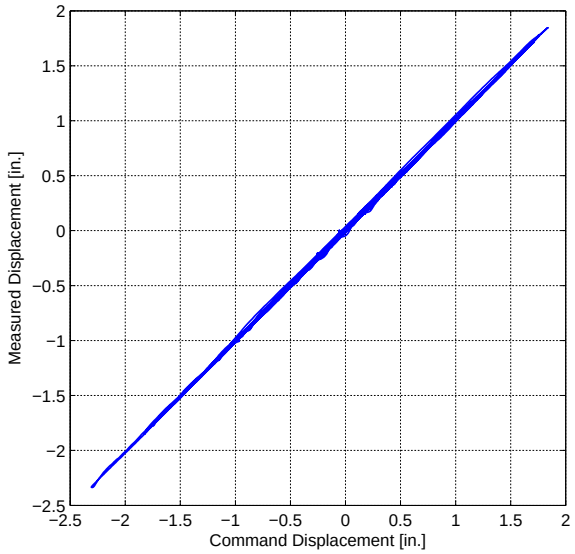


Fig. 6.13 Synchronization subspace plot from STS.

The plot presented here of the real-time test shows an almost perfectly straight line turned slightly counter-clockwise. This means that the actuator was overshooting a little bit and confirms the conclusions, which were made by the investigation of energy dissipation. As can be seen from Fig. 6.14, there is three distinct amplitude peaks in the Fourier spectrum of the measured resisting forces. The first peak at a frequency of 1.53 Hz again resembles the fundamental frequency of the one-bay frame model. However, the second peak, which occurred around a frequency of 37 Hz, corresponds to the natural frequency of the reaction frame and was therefore generated by the interaction of the test specimen with the reaction frame and the

actuator. The third amplitude peak at a frequency of 148 Hz corresponds again to the oil column frequency of the μNEES setup and thus confirms once more the analytically estimated value.

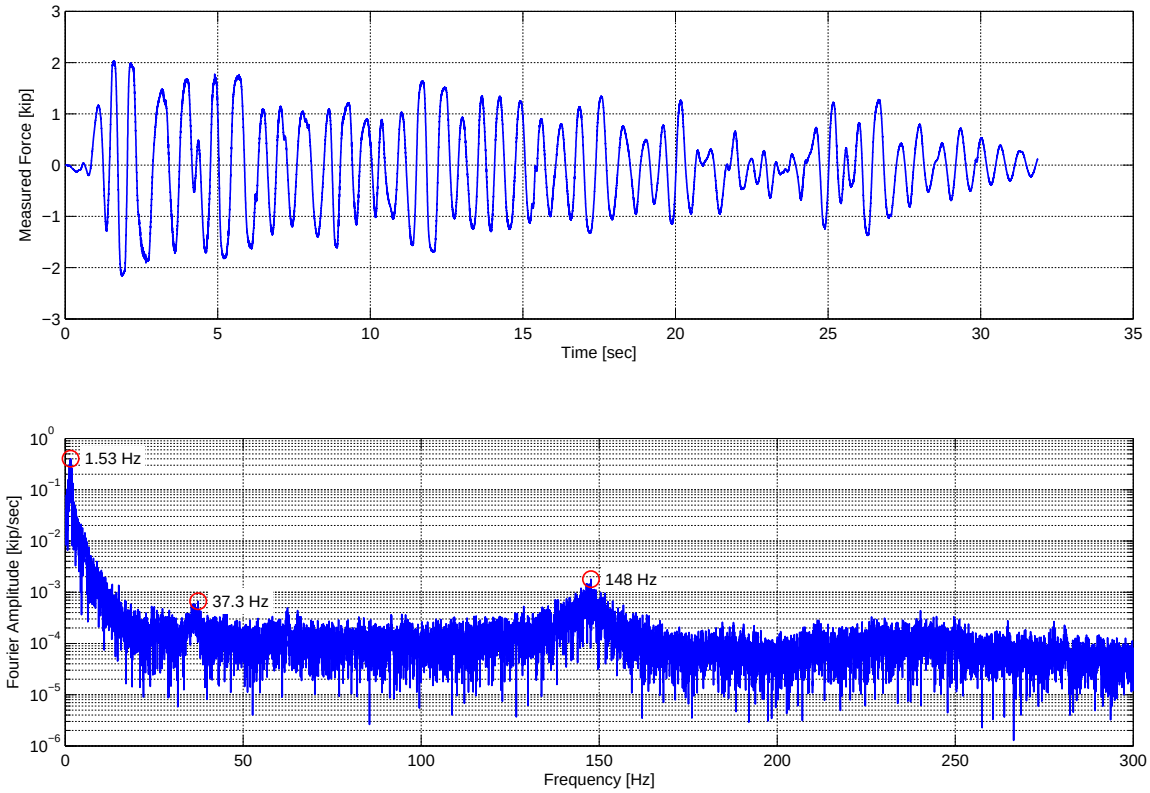


Fig. 6.14 Measured force history and Fourier amplitude spectrum from STS.

Comparison of Real-Time versus Slow Hybrid Simulation:

In order to determine if the structural response of the one-bay frame model was affected by the testing rate of the hybrid simulation, a second, ten-times slower, local hybrid simulation was carried out. The simulation time step was increased to Δt_{sim} = 200.2 msec, which effectively yielded a hybrid simulation that executed approximately ten-times slower than the soft real-time geographically distributed one. As can be seen from Fig. 6.15 the displacement, velocity, and acceleration response at the top of the left experimental column (node 3) were almost identical. This means that the effect of the testing rate for this specific test and structure was negligible and can be ignored. The comparison of the hysteresis loops and the experimental shear forces in Fig. 6.16 confirm this observation. Because the response of the one-bay-frame structure was barely affected by the rate of testing, two conclusions can be drawn. First, the behavior of the two ductile steel coupons, simulating the plastic hinge zone at the end of the column, was independent of strain rates.

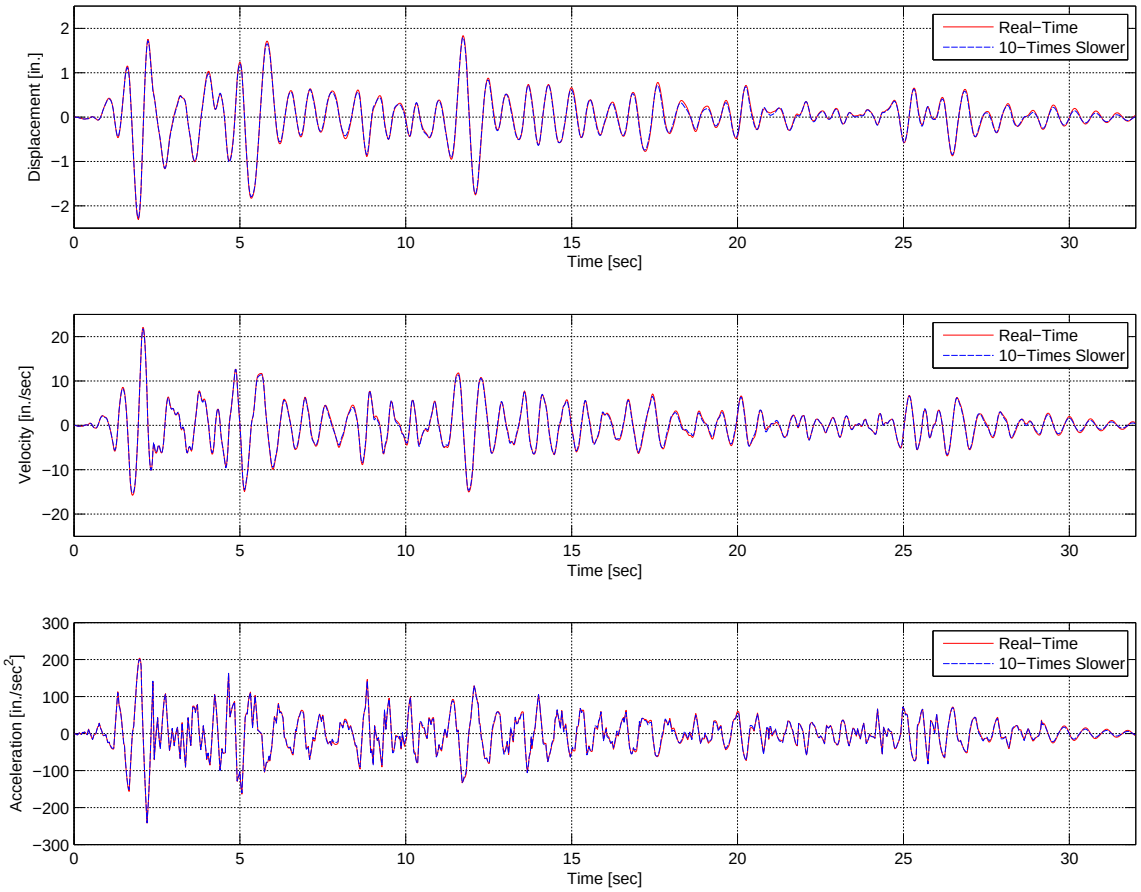


Fig. 6.15 Comparison of response histories of one-bay-frame model for real-time versus slow hybrid simulation.

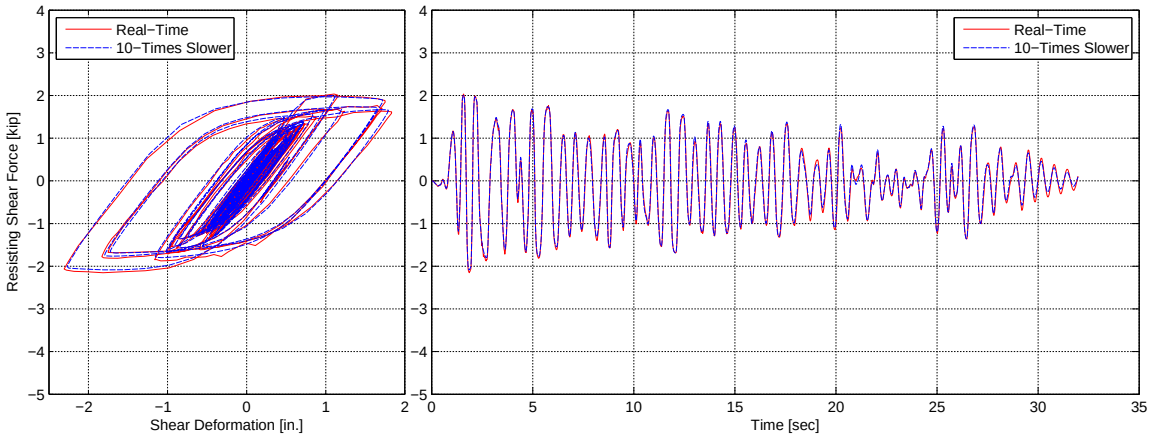


Fig. 6.16 Comparison of hysteresis loops and force histories of one-bay-frame model for real-time versus slow hybrid simulation.

Secondly, the effects of the inertial forces created by the physical mass of the specimen and loading apparatus were small compared to the ones generated by the mass of the model at node 3. It was determined that the maximal physical inertial force measured by the load cell and fed back into the analysis was approximately 0.5% of the analytical inertial force at node 3, which confirms the second conclusion.

To compare the performance of the servo-hydraulic control system achieved during the real-time and slow tests, a tracking indicator as defined by Mercan (2007) was utilized. The tracking indicator is based on the enclosed area of the hysteresis loops when measured displacements are plotted versus command displacements, as in Fig. 6.13. If the value of the indicator is increasing, the displacement response is leading and consequently introducing additional energy dissipation (damping) into the system. On the other hand, if the indicator is decreasing, the displacement response is lagging and the energy dissipation due to those tracking inaccuracies is negative. If the energy introduced into the system by this systematic experimental error is larger than the sum of the hysteretic, viscous, and numerical energy dissipation, the response of the system starts growing, ultimately yielding the system unstable.

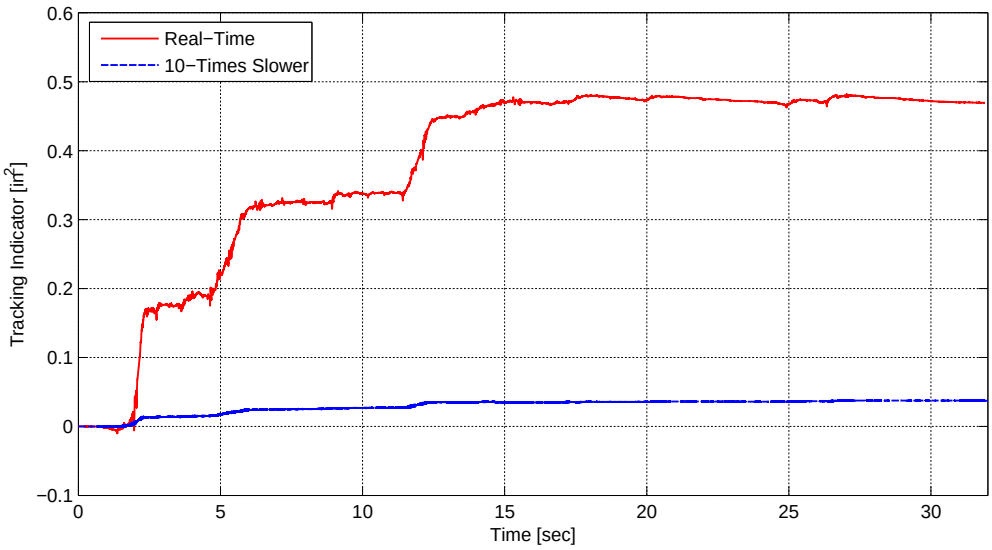


Fig. 6.17 Comparison of tracking performance for real-time versus slow hybrid simulation.

Fig. 6.17 presents the tracking indicators for the two tests, where the slow test was compressed in time by a factor of ten to facilitate easier comparison. From the positive values of the two tracking indicators it can be inferred that additional damping was introduced into the

system during both hybrid simulations. Even though this affected the accuracies of the tests this experimental error was preferable as opposed to the systematic introduction of additional energy, which could have destabilized the system. Although tracking was good for both tests, it can be observed that the performance during the slow test was approximately 12 times better than during the real-time test.

Comparison of Analytical Simulation versus Hybrid Simulation:

To validate the purely analytical OpenSees models of the μNEES specimen and the one-bay-frame structure, the numerical results were compared against the experimental results obtained from the quasi-static test and the hybrid simulation. The analytical model of the μNEES specimen consisted of several linear elastic elements resembling the clevis and the S4x7.7 steel column and four nonlinear force beam-column elements modeling the two steel coupons.

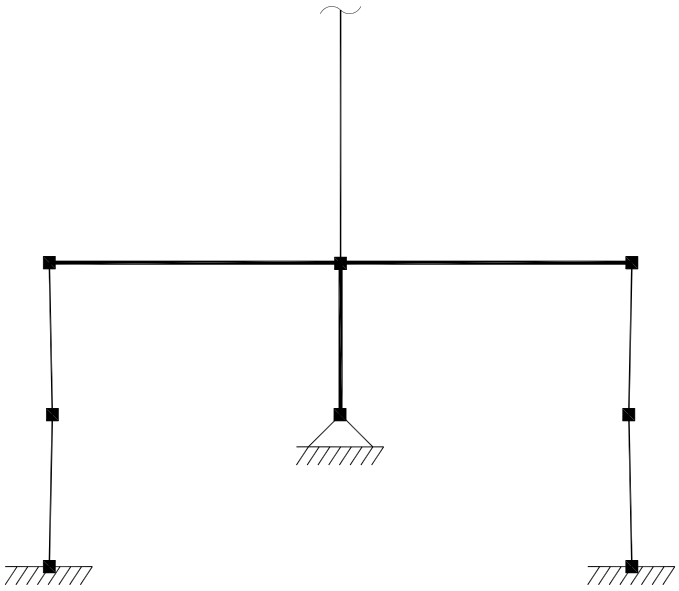


Fig. 6.18 Analytical model of μNEES specimen.

In order to capture the buckling behavior of the steel coupons an approach introduced by Uriz (2005) was adopted, where the exact large-displacement corotational geometric-transformation was utilized in combination with an initial offset (perpendicular to the coupon's local x-axis) of the center node of the coupon. For each nonlinear force beam-column element five Gauss-Lobatto integration points with circular fiber cross-sections were used and all the fibers were modeled utilizing the Menegotto-Pinto plasticity model without isotropic hardening.

In a first step the parameters of the analytical model of the μNEES specimen were calibrated using the experimentally determined hysteresis loops from the SAC loading history as shown in Fig. 6.4. As can be seen from Fig. 6.19, the calibrated analytical model was capable of capturing the overall response quite accurately. In addition it can be observed that the strength reduction with each cycle, due to the buckling of the coupons, was accurately modeled by the nonlinear force beam-column elements and the corotional geometric-transformations. For positive displacements the match between the numerically generated hysteresis loops and the experimentally generated ones was slightly better than for negative displacements.

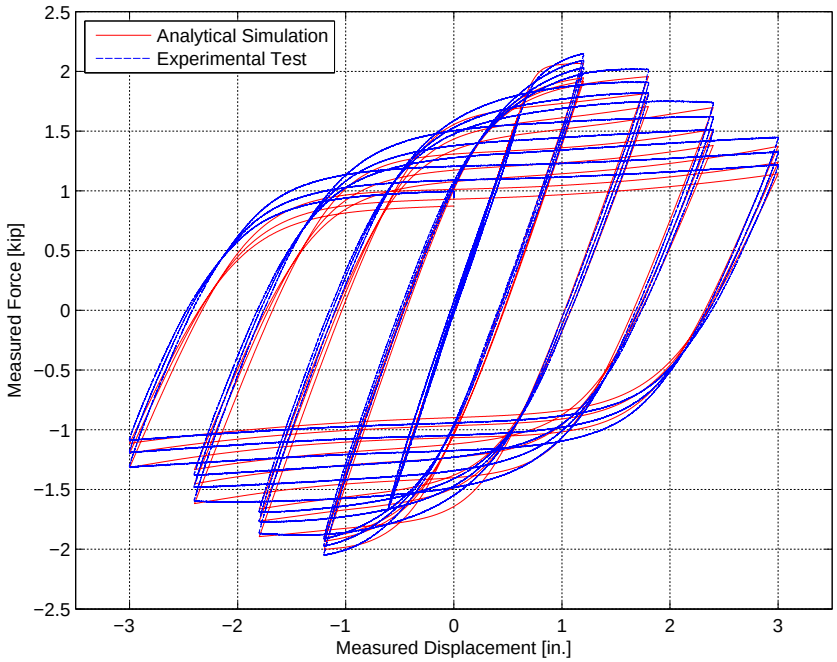


Fig. 6.19 Comparison of analytical versus experimental hysteresis loops for SAC loading history.

After the successful calibration of the numerical specimen, the subassembly was incorporated into a complete analytical model of the one-bay-frame structure in order to perform the dynamic direct integration analysis. The earthquake response of the purely analytical structure was then compared against the results obtained from the hybrid simulation test. As can be seen from Fig. 6.20 the displacement, velocity, and acceleration response at the top of the left column (node 3) were nearly identical for the analytical and the hybrid model. This means that the purely numerical model was capable to accurately capture the earthquake response of the

nonlinear one-bay-frame structure. The comparison of the hysteresis loops and the shear forces of the left column are shown in Fig. 6.21. It can be seen that the column shear forces computed by the analytical model matched well with the shear forces measured from the experimental column specimen. However, the resisting forces were captured with slightly less accuracy than the displacements, velocities, and accelerations. This can also be observed from the hysteresis loops where the numerical model of the specimen exhibited too much softening in the second response cycle as compared to the experimental specimen.

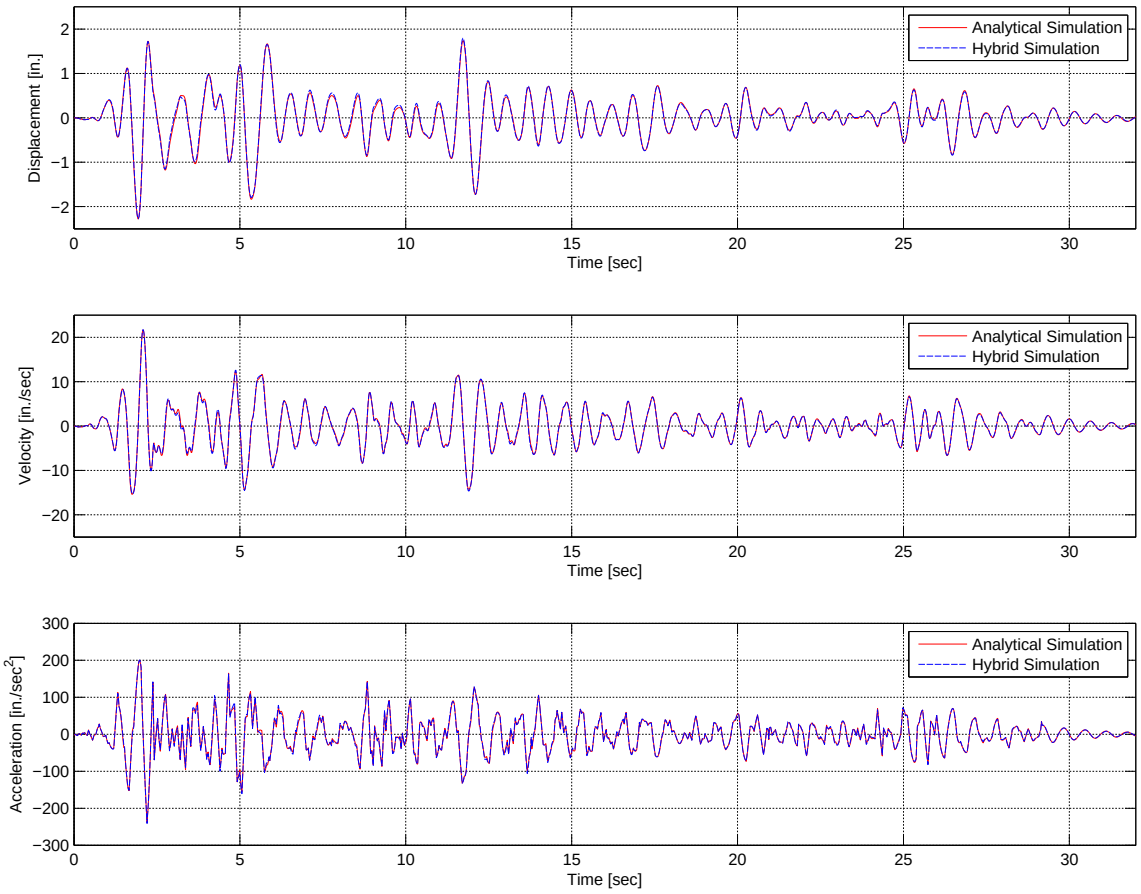


Fig. 6.20 Comparison of response histories of one-bay-frame model for analytical versus hybrid simulation.

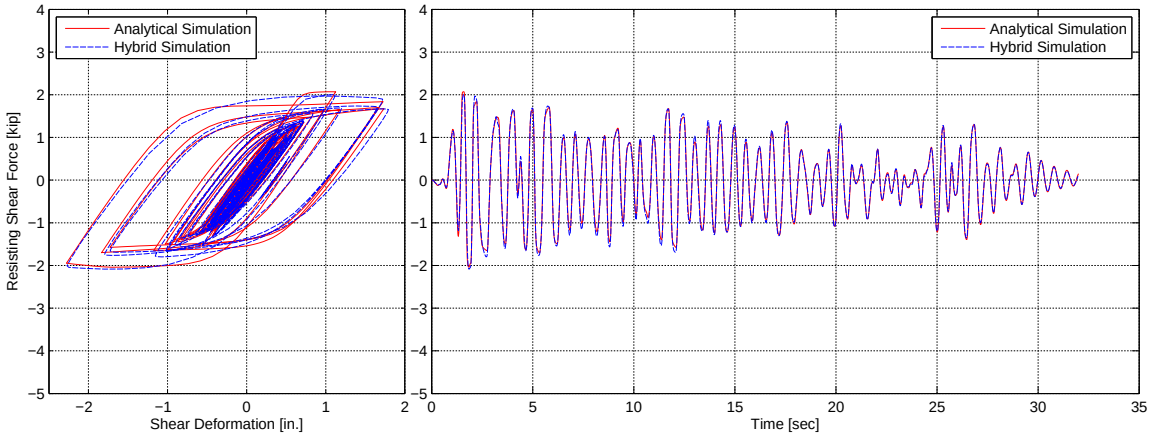


Fig. 6.21 Comparison of hysteresis loops and force histories of one-bay-frame model for analytical versus hybrid simulation.

Nevertheless, the overall response of the purely analytical and the hybrid structure matched very well, which successfully demonstrated the validity of both models. Particularly, it proved that the geographically distributed hybrid simulation is a viable testing method that can be utilized to conduct soft real-time experimental tests among different laboratories by relying on the next-generation Abilene network.

6.3 STRUCTURAL COLLAPSE SIMULATION

6.3.1 Motivation

One important research area in the field of earthquake engineering is concerned with the ability to model structural collapse in order to improve our understanding of the performance of structures during very rare earthquake events. However, experimental investigations of collapse are in general complex, expensive, and potentially dangerous. Any of the three previously described experimental methods (see Chapter 2) can technically be employed to perform structural collapse tests. However, quasi-static and shaking table tests both face difficulties in setting up, executing, and interpreting such tests, whereas hybrid simulation does not.

The advantages and disadvantages of the three experimental methods with respect to structural collapse investigations are summarized next.

1. While quasi-static tests are relatively easy and economical to perform, the demands imposed on the test specimens are not directly related to the observed damage. This raises

questions about whether the specimens are under- or overtested, which in turn makes it extremely difficult to draw accurate conclusions about the behavior of a structure near or at collapse.

2. Shaking table tests provide important data on the dynamic response to specific ground motions considering the inertial and energy-dissipation characteristics of the structure and the consequences of geometric nonlinearities, localized yielding and damage on the response. However, for such tests, a complete structural system is required, and the specimen needs to be constructed including complex active or passive gravity load setups. Furthermore, preventive measures need to be taken to protect expensive test equipment from specimen impact during collapse. This makes investigations of collapse on shaking tables very complex, expensive, and potentially dangerous. In addition, due to simulator platform and/or control system limitations, reduced-scale or highly simplified specimens are commonly necessary, which call into question the realism of many shaking table tests of structural collapse.
3. Hybrid simulations make it possible to investigate geometric nonlinearities, three-dimensional effects, multiple-support excitation, and soil-structure interactions by incorporating them into the analytical portion of the hybrid model. Hence, the hybrid simulation method gives the researcher the ability to test large- or full-scale specimens and to reduce or avoid scaling issues. In addition, only the critical, collapse-sensitive elements of the structure need to be physically tested, allowing for a substantial increase in the number of different collapse tests afforded by the same budget.

To demonstrate how gravity loads and geometric nonlinearities can be accounted for in the numerical part of the hybrid model, a hybrid simulation, wherein a portal-frame is tested by consistently accounting for second-order effects due to gravity loads, is carried out using the Open System for Earthquake Engineering Simulation (OpenSees) (McKenna 1997; Fenves et al. 2004) and the OpenFresco (see Chapter 3) software packages.

6.3.2 Second-Order Effects

Because second-order effects are caused by gravity loads, it is important to understand how such loads affect the response of a structure. In general, gravity loads influence the response of a structure on three levels, which can be categorized as shown in Table 6.5 below.

Table 6.5 Effects of gravity loads on the structural response.

Level	*Effect*
Section	Gravity loads cause axial forces that interact with bending moments and shears at the cross section, commonly known as P-M and P-V interaction.
Element	Gravity loads affect the stability of structural elements and critical regions, causing softening and local and/or lateral-torsional buckling of elements, commonly known as the small P-Delta (P-δ) effects.
Structure	Gravity loads influence the stability of a structure, causing additional overturning moments due to large displacements. These effects are commonly known as the large P-Delta (P-Δ) effects.

To model these nonlinear geometric effects in finite element software (such as OpenSees), the element equilibrium equations need to be satisfied in the deformed configuration. Furthermore, because of large displacements, the compatibility relations between element deformations and global element end-displacements become nonlinear. For frame elements, it is then possible to separate these coordinate transformations from the actual implementation of the element itself as long as the element force-deformation relations are defined in a basic coordinate system without rigid body modes. Hence, different geometric theories can be implemented for the same element without modifying the element code. In OpenSees the *CrdTransf* software class provides this exact functionality by transforming the response quantities of any frame element from the global degrees of freedom to the simply supported basic element degrees of freedom and back. The *CrdTransf* class is an abstract base class that has concrete subclasses for linear, p-delta, and corotational geometric transformations. This abstraction and encapsulation of the frame element coordinate transformations facilitates the switching among different linear and nonlinear coordinate transformations in the OpenSees software framework without affecting the implementations of the frame elements themselves.

6.3.3 Theory and Implementation

The 2D experimental beam-column element formulation in OpenFresco (which is used in this case study) is based on the three collocated degrees-of-freedom of the cantilever basic system (see Fig. 6.22 or Chapter 3).

The necessary transformations of displacements, velocities, accelerations, and forces are implemented in the concrete EEBeamColumn2d class. Since this experimental element is a frame member similar to an analytical beam-column element, the previously described OpenSees *CrdTransf* class is employed to transform the response quantities from the global degrees of freedom to the simply supported basic element degrees of freedom and back. However, because the simply supported basic system is not well suited for experimental testing (due to the two rotational degrees of freedom), the response quantities are further transformed from the simply supported to the cantilever basic system and back. These additional nonlinear transformations are directly implemented in the EEBeamColumn2d class and described in detail in Chapter 3.

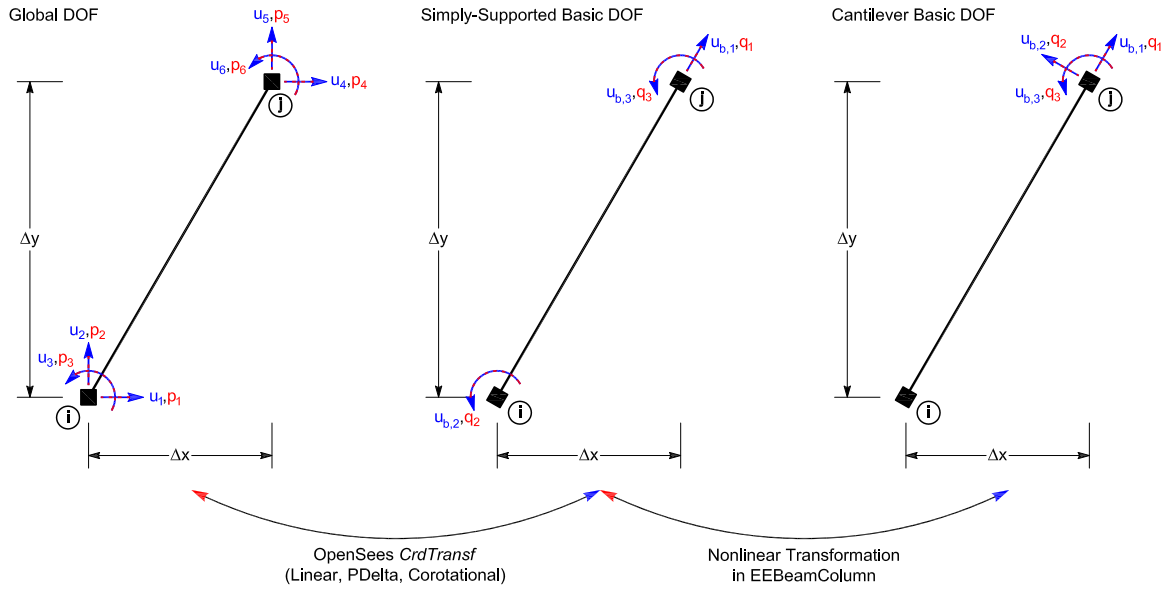


Fig. 6.22 Experimental beam-column element (EEBeamColumn2d).

Of the three coordinate transformations (linear, p-delta, and corotational) that are available in OpenSees, only the large-displacement, moderate deformations corotational transformation (de Souza 2000; Filippou and Fenves 2004) is explained here.

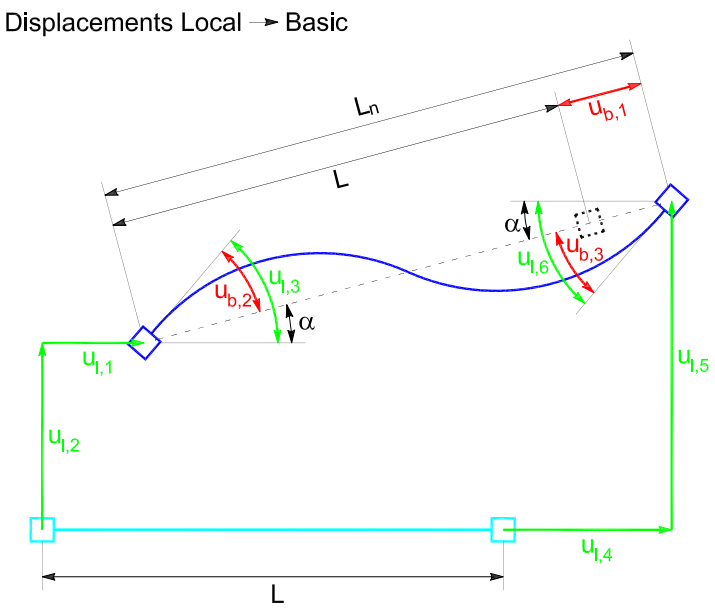


Fig. 6.23 Corotational transformation from local to Basic System A.

In a first step (which is not shown), the six end-displacements of the beam-column element are transformed from the global to the local coordinate system by applying a rotational transformation matrix. Afterwards, the element trial displacements, u_l, in the local coordinate system are transformed to the element deformations, u_b, in the basic coordinate system using.

$$\begin{aligned} u_{b,1}^{(A)} &= L_n - L \\ u_{b,2}^{(A)} &= u_{l,3} - \alpha \\ u_{b,3}^{(A)} &= u_{l,6} - \alpha \end{aligned} \qquad with \qquad \begin{aligned} L_n &= \sqrt{\left(L + u_{l,4} - u_{l,1}\right)^2 + \left(u_{l,5} - u_{l,2}\right)^2} \\ \alpha &= \arctan\left(\frac{u_{l,5} - u_{l,2}}{L + u_{l,4} - u_{l,1}}\right) \end{aligned} \tag{6.1}$$

where L_n is the length of the element chord in its deformed configuration and α is the rotation of the chord from the undeformed to the deformed element configuration. As previously shown in Chapter 4 and repeated here for convenience, the transformations of the resisting forces and the element stiffness matrix from the Basic System A to the local coordinate system take the following form.

$$\begin{aligned} \mathbf{p_l} &= \mathbf{a}^T \mathbf{q}^{(A)} \\ \mathbf{k_l} &= \frac{\partial \mathbf{p_l}}{\partial \mathbf{u_l}} = \frac{\partial}{\partial \mathbf{u_l}}\left(\mathbf{a}^T \mathbf{q}^{(A)}\right) = \mathbf{a}^T \mathbf{k_b}^{(A)} \mathbf{a} + \frac{\partial \mathbf{a}^T}{\partial \mathbf{u_l}} \mathbf{q}^{(A)} \end{aligned} \tag{6.2}$$

where the compatibility matrix, **a**, and the elastic element stiffness matrix, $\mathbf{k_b}^{(A)}$, in the basic coordinate system A are given in Equation (6.3).

$$\mathbf{a} = \begin{bmatrix} -\cos(\alpha) & -\sin(\alpha) & 0 & \cos(\alpha) & \sin(\alpha) & 0 \\ -\frac{\sin(\alpha)}{L_n} & \frac{\cos(\alpha)}{L_n} & 1 & \frac{\sin(\alpha)}{L_n} & -\frac{\cos(\alpha)}{L_n} & 0 \\ -\frac{\sin(\alpha)}{L_n} & \frac{\cos(\alpha)}{L_n} & 0 & \frac{\sin(\alpha)}{L_n} & -\frac{\cos(\alpha)}{L_n} & 1 \end{bmatrix} \tag{6.3}$$

$$\mathbf{k}_{\mathbf{b}}^{(A)} = \begin{bmatrix} \frac{EA}{L} & 0 & 0 \\ 0 & \frac{4EI}{L} & \frac{2EI}{L} \\ 0 & \frac{2EI}{L} & \frac{4EI}{L} \end{bmatrix}$$

$\mathbf{u_l}$ and $\mathbf{p_l}$ are the element displacements and forces in the local coordinate system, $\mathbf{u_b}^{(A)}$ and $\mathbf{q}^{(A)}$ are the element deformations and forces in the basic coordinate system A, α is the chord rotation, and L_n is the length of the element chord in its deformed configuration. Furthermore, the first part of the local stiffness matrix in (6.2) represents the material stiffness, and the second part represents the geometric stiffness.

6.3.4 Test Structure and Earthquake Record

This section demonstrates how the novel experimental beam-column elements (EEBeamColumn2d; see Chapter 3) can be used to perform a hybrid simulation until collapse. As can be seen from Fig. 6.24 the steel single-story, single-bay portal-frame had a height of 50 inches, which corresponds to the height at which the actuators attached to the experimental steel specimen. The width of the structure was arbitrarily chosen to be 100 inches. Furthermore, the portal-frame consisted of an elastic W6x12 beam and two ductile, nonlinear S4x7.7 columns. The connecting beam, the viscous energy-dissipation and mass properties as well as the gravity and earthquake loads were all analytically modeled in OpenSees. The EEBeamColumn2d experimental element (see Fig. 6.22) was used to represent both columns. The left column was simulated using the ECSimDomain controller and the right column was physically tested in the μNEES laboratory.

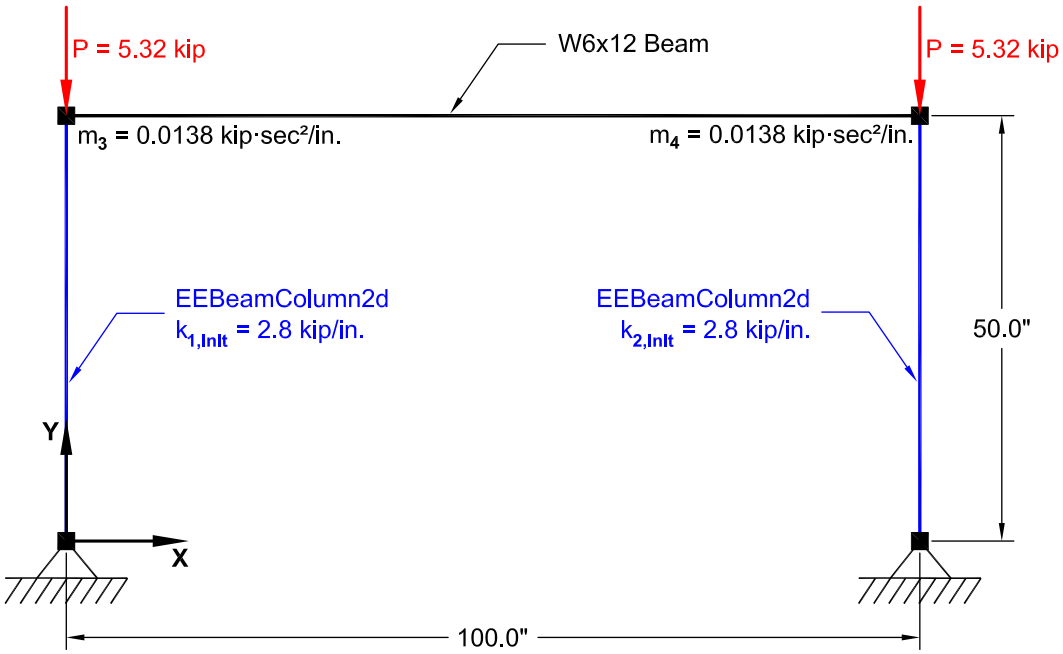


Fig. 6.24 Portal-frame model with structural properties.

Considering the axial and flexural deformations in the elements, the portal-frame model had a total of eight degrees of freedom (four translational and four rotational). However, it was assumed that only the four translational degrees of freedom had mass ($m_3 = m_4 = 0.0138$ kip·sec^2/in.), which made the 8x8 mass matrix singular with rank equal to four. With the selected structural properties, the fundamental period of the structure was 0.49 sec, and the viscous energy dissipation was modeled with 5% mass proportional damping. The magnitude of the gravity loads (P = 5.32 kip) was set at 50% of the critical side-sway buckling load of the physical column (including the clevis) obtained by assuming that the beam was flexurally rigid and the base support of the column was pinned. Because the gravity loads on the two columns were entirely treated in the analytical portion of the hybrid model, with negligible axial loads imposed on the experimental parts of the structure (column cross-sections and elements), only the large p-delta effects (see Table 6.5) are accounted for in this demonstration and validation example.

The hybrid portal-frame model was subjected to the SACNF01 near-fault ground motion of the SAC database, scaled to a peak ground acceleration of 0.906 g. The ground motion was originally recorded during the 1978 Tabas, Iran, earthquake and modified based on the SAC protocol to represent NEHRP soil type Sd. The 30 sec time history was shortened to 25 sec so that the hybrid simulations took less time to execute. It therefore consisted of 2500 data points

recorded every 0.01 sec. The elastic response spectra of such motion for a damping ratio of 5% and with markers at the fundamental side-sway period are shown in Fig. 6.25.

6.3.5 Test Setup and Procedure

For convenience in the laboratory, the right beam-column specimen was inverted with the pin end at the top and the fixed end at the bottom. The μNEES experimental setup, which was described in detail in Section 6.2.3, was used to test the right column of the portal-frame. It is important to emphasize that because the basic coordinate system moved as the element deformed, the one-actuator experimental setup (Fig. 6.3) utilized in these tests also behaved as if it had been attached to and moved with the experimental element. Because the tests were performed as local hybrid simulations, the OpenSees finite element analysis software in combination with OpenFresco's tightly integrated experimental element, local experimental site, experimental setup, and control objects were all located at the *nees@berkeley* laboratory. Due to the fact that only two of the eight degrees of freedom had mass assigned an unconditionally stable integration method had to be used to solve the equations of motion.

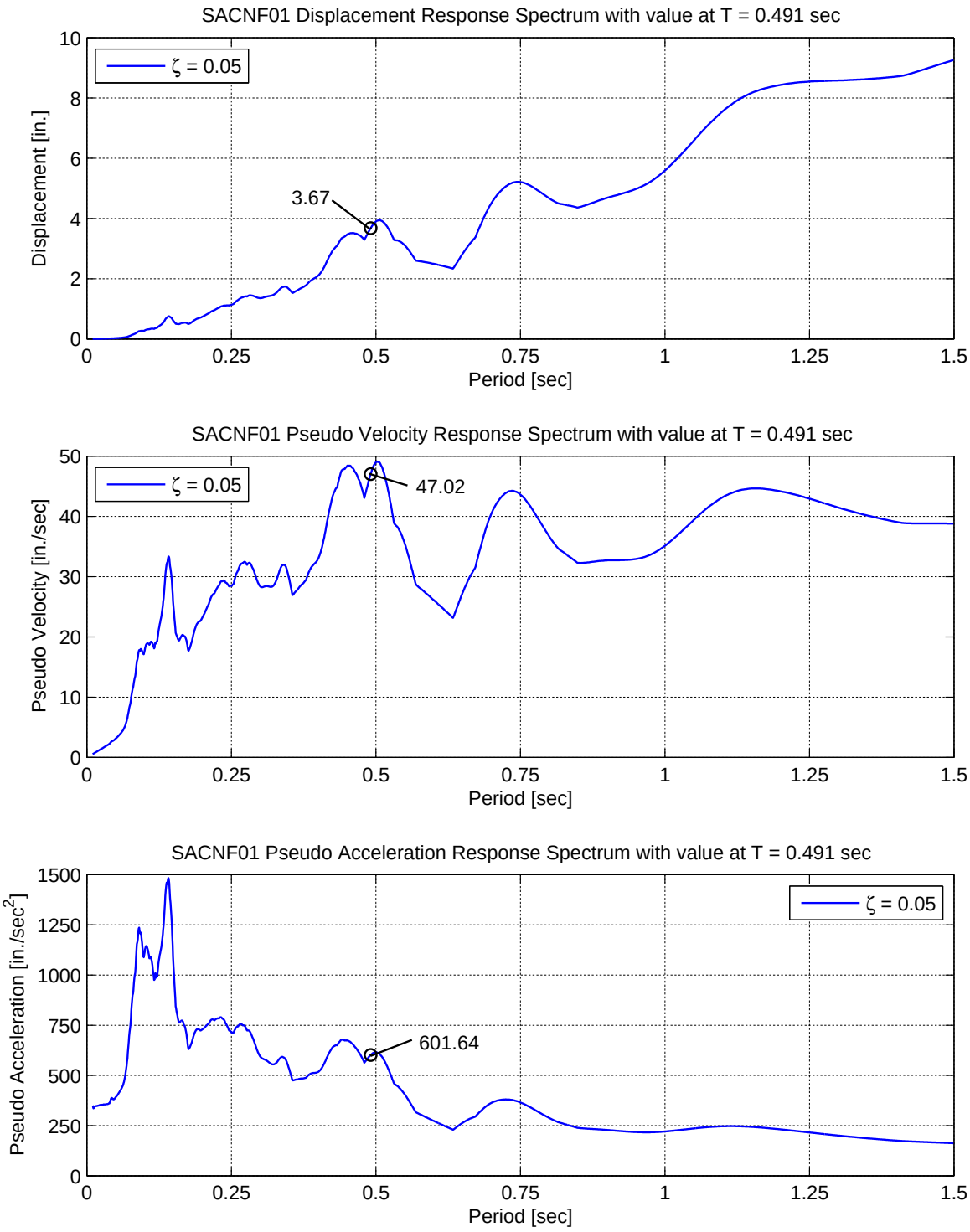


Fig. 6.25 Elastic response spectra for SACNF01 ground motion, with values at fundamental period.

In addition, since the hybrid model consisted of both numerical and experimental elements that exhibited strong material and geometric nonlinearities, the generalized-alpha-OS method (see Chapter 4) was not well suited and an iterative solution scheme had to be employed instead. Hence, the second modified implicit Newmark method with 10 sub-steps and the Newton-Raphson algorithm with a mixed Jacobian were utilized (see Chapter 4).

As with the previous case study, the OneActuator experimental setup in OpenFresco was utilized to represent the μNEES setup and the xPCtarget experimental control object was employed to connect to the real-time xPC Target machine, which was running the event-driven, third-order predictor-corrector algorithm that uses the last predicted value in the corrector formula (see Chapter 5). The integration time step was chosen to be $\Delta t_{int} = 0.01$ sec, which was equal to the ground motion time step. On the other hand, the simulation time step was chosen to be $\Delta t_{sim} = 2^{-5} = 31.25$ msec and in combination with the 10 iterations per time step, the hybrid simulation was performed at a rate 31.25 times slower than real time, and each test lasted 13 minutes.

6.3.6 Test Results and Interpretation

To investigate the effect of gravity loads on the response of the portal-frame, two pairs of simulations were performed. One simulation was performed without any gravity loads applied, and then gravity loads were added to the analytical portion of the hybrid model for the second test. Both tests utilized the corotational geometric coordinate transformation and identical new coupons were used in the clevis representing the plastic hinge in each test.

Various results are described in this section. Firstly, the response that was recorded within OpenSees is discussed. Secondly, some predictor-corrector results that were recorded on the xPC Target machine are presented. The measurements recorded by the STS controller software are then used to evaluate the performance of the servo-hydraulic control system that was achieved during the test. Finally, the hybrid simulations are compared against purely analytical simulations of the portal-frame model.

Finite Element Model Results:

The story-drift time histories as well as the story-shear vs. story-drift hysteresis loops are compared in Fig. 6.26. Even though the frame drifts and the residual displacements were

substantial for the first test without gravity loads, the hybrid model did not collapse. In the second hybrid simulation, the presence of the gravity loads is clearly evident from the negative post-yield tangent stiffness exhibited by the combined hybrid model.

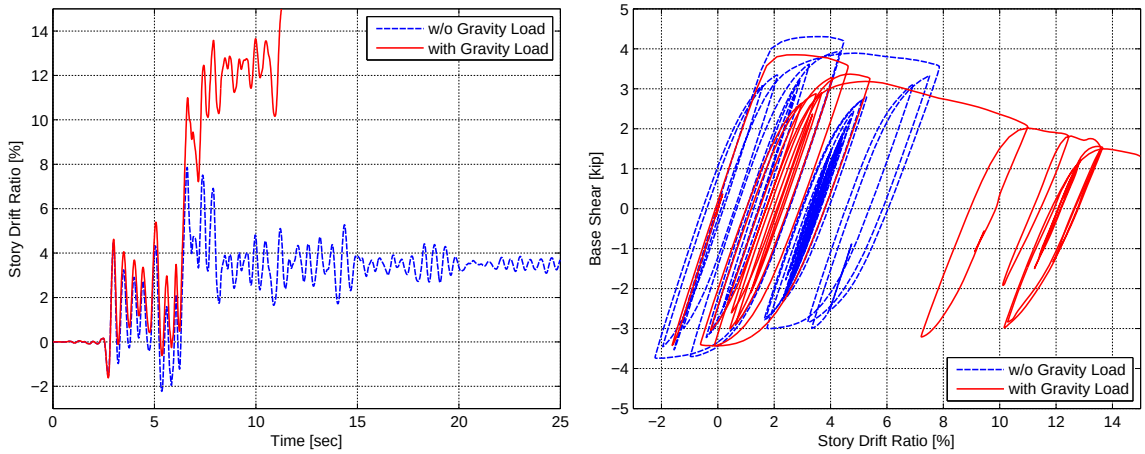


Fig. 6.26 Comparison of story-drift time-histories and total story hysteresis loops for portal-frame model from OpenSees.

As can be seen from Fig. 6.26a, for the hybrid simulation with gravity loads, the story drifts increased rapidly at 6.5 sec into the test, which can be considered as the initiation of frame collapse. With each additional cycle, the portal-frame leaned over further until the hybrid simulation was finally terminated when the actuators reached their stroke limit of 7.5 inches, which corresponded to an interstory drift ratio of 15%. The second simulation demonstrated that the hybrid model could be tested all the way to collapse of the portal-frame, which would be difficult and dangerous to accomplish in shaking table tests.

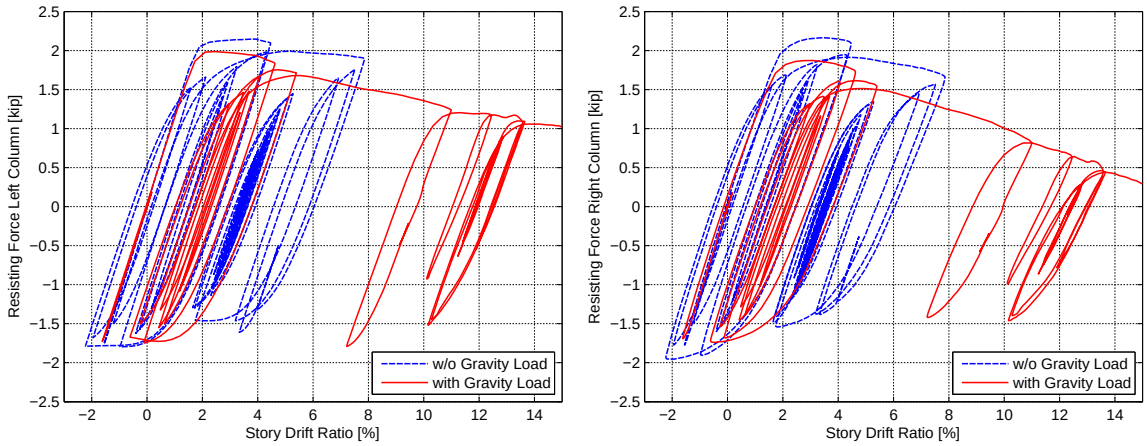


Fig. 6.27 Comparison of element force-story-drift hysteresis loops for portal-frame model from OpenSees.

Fig. 6.27 provides the comparison of the hysteresis loops of the two experimental column elements. The element shear forces are plotted against the element drift ratios in the global reference coordinate system. It can be observed that for both hybrid simulations (the one without and the one with gravity loads) the negative post-yield stiffnesses of the right column were significantly larger than the ones of the left column. This behavior can be explained by the fact that the portal-frame was leaning to the right and the overturning moment was therefore loading the right column further in compression and unloading the left column. Since both hybrid simulations utilized large-displacement nonlinear geometric coordinate transformations for all the elements, this effect could also be observed in the test without the gravity loads.

Fig. 6.28 compares the actuator hysteresis loops for the two experimental beam-column elements. The actuator-force vs. actuator-displacement relationships are plotted in the Basic System B (cantilever) as they were acquired from the ECSimDomain (left column) and the ECxPCtarget (right column) control objects. This means that the responses are compared before they were transformed to the Basic System A (simply supported) and before the gravity load effects were applied by the nonlinear geometric transformations in the EEBeamColumn objects. Thus, no negative post-peak stiffnesses were observed in the actual test. The differences between the hysteresis loops are entirely due to the consistently applied displacements. Furthermore, it can be seen that for large displacements of the cantilever column ends (respectively large rotations of the clevises), the hysteresis loops exhibited some degradation. This degradation was due to the buckling of the replaceable steel coupons as the clevises went through large rotations.

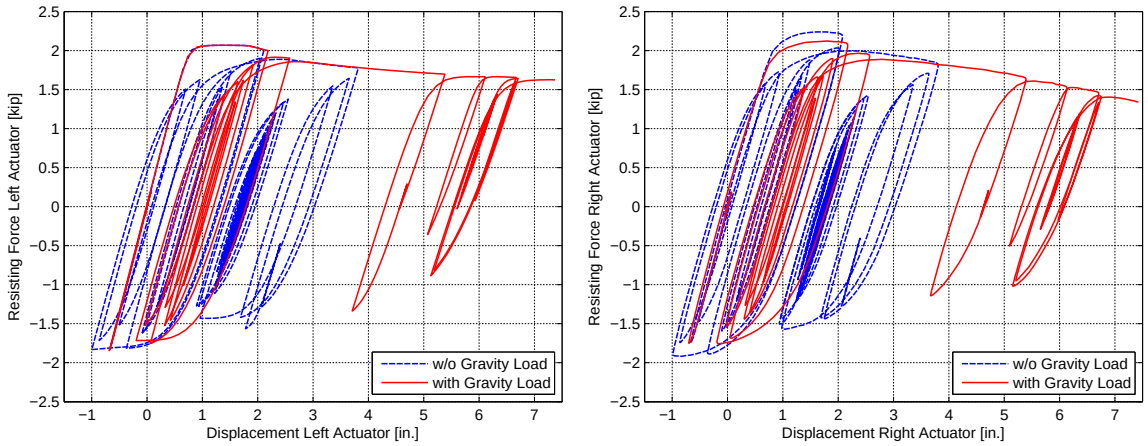


Fig. 6.28 Comparison of actuator force-displacement hysteresis loops for portal-frame model from OpenSees.

Predictor-Corrector Results:

Similarly to the previous case study, the correct operation of the predictor-corrector finite state machine is again confirmed by taking a closer look at the displacement command generation.

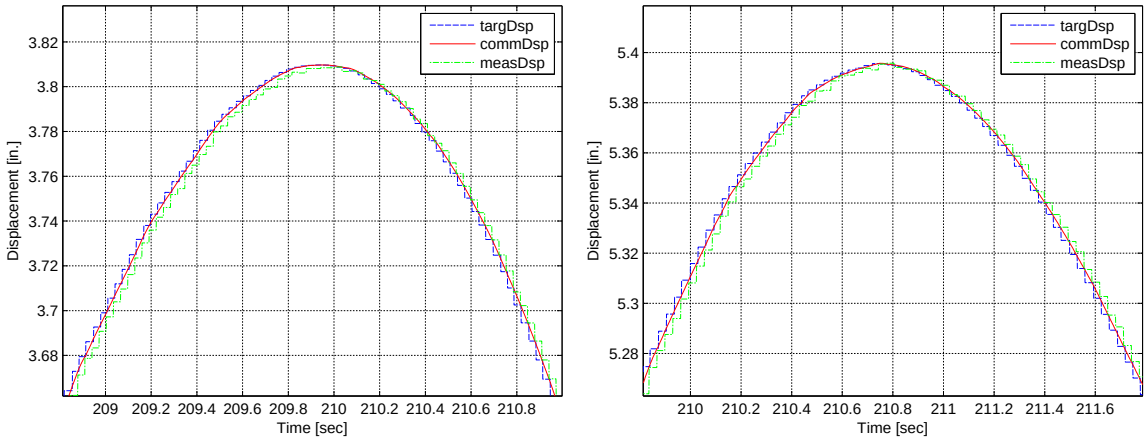


Fig. 6.29 Close up of predictor-corrector state transitions from xPC Target.

The two graphs in Fig. 6.29 show close-ups of the signals for the test without gravity loads (left) and the one with gravity loads (right). The smoothness of the actuator displacement command signals (solid red lines), which have no jumps and kinks, verify that the chosen third-order predictor-corrector algorithm that utilizes the last predicted value in the corrector formula (see Chapter 5), achieved satisfactory performance.

Furthermore, it can be observed that due to small tracking errors, the measured displacement signals (dash-dotted green lines) did not always exactly reach the intersection of

the target displacement signals (dashed blue lines) with the command displacement signals (solid red lines). Particularly, the measured displacement signals were a little smaller than the commanded ones. This means that the actuators were slightly undershooting, which is confirmed from the results acquired by the STS control software in the next section.

Servo-Hydraulic Control System Results:

The performance of the servo-hydraulic control system is evaluated from the displacement and force measurements, which were acquired by the STS control software at a sampling rate of 1024 Hz. For the first hybrid simulation without gravity loads, the largest displacement error during the test was equal to 0.006 in., which corresponded to approximately 0.15% of the maximal displacement; and for the second hybrid simulation with gravity loads, the largest displacement error during the test was equal to 0.005 in., which corresponded to approximately 0.06% of the maximal displacement. This means that the servo-hydraulic control system was able to achieve very accurate displacement tracking, mainly due to the slow execution of the tests (31.25-times slower than real time).

Actuator lag, lead, undershoot, and overshoot were again evaluated from synchronization subspace plots (Mercan 2007), where the measured displacements are plotted against the command displacements. The following actuator performances can quickly be determined from the graph: If the plot is a straight 45º-line perfect tracking without any error was achieved. If the plot shows a counterclockwise or clockwise turning ellipsoid, the actuator produced a lagging or leading displacement response. Finally if the line or the axis of the ellipsoid is turned clock-wise or counter-clock-wise away from the 45º-line, the actuator was undershooting or overshooting.

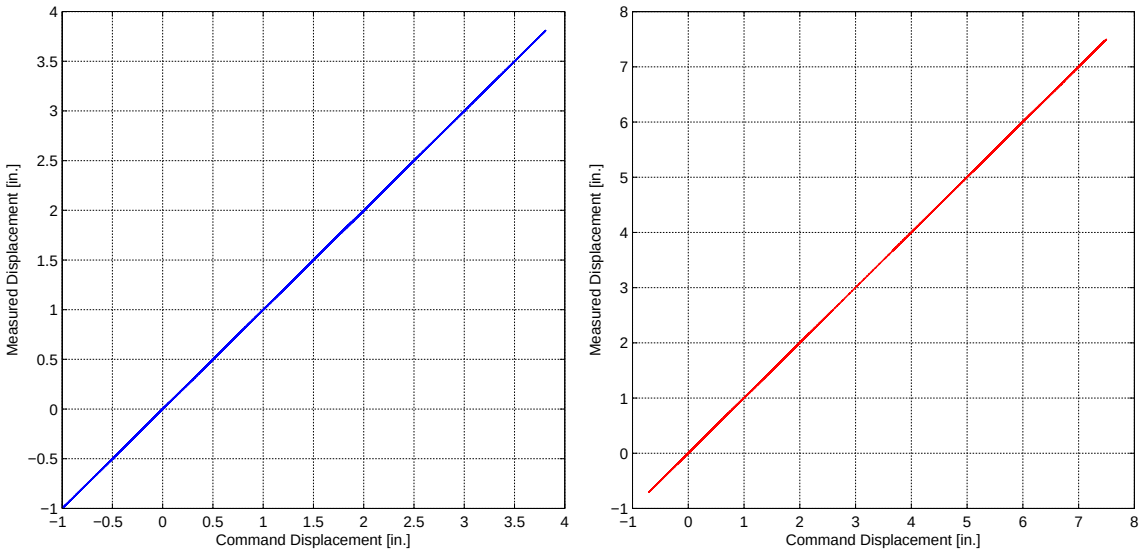


Fig. 6.30 Synchronization subspace plots from STS.

The two plots for the hybrid simulations without gravity loads (left) and with gravity loads (right), presented in Fig. 6.30, show almost perfectly straight lines, which are minutely turned clock-wise. This means that during both tests the actuator was undershooting a little bit, introducing a minuscule amount of additional energy into the system. Furthermore, this confirms the observations that were made in the previous section.

Fig. 6.31 presents the tracking indicators for the two tests (Mercan 2007). From the negative values of the two indicators it can be inferred that negative damping, meaning additional energy, was introduced into the system during both hybrid simulations. However, the values of the tracking indicators are so small that only minuscule amounts of energy were introduced into the hybrid simulations, which did not affect the stability of the system. Although tracking was excellent for both tests, it can be observed that tracking during the test with gravity loads was slightly better than tracking during the tests without gravity loads but with more noise in the measured signals.

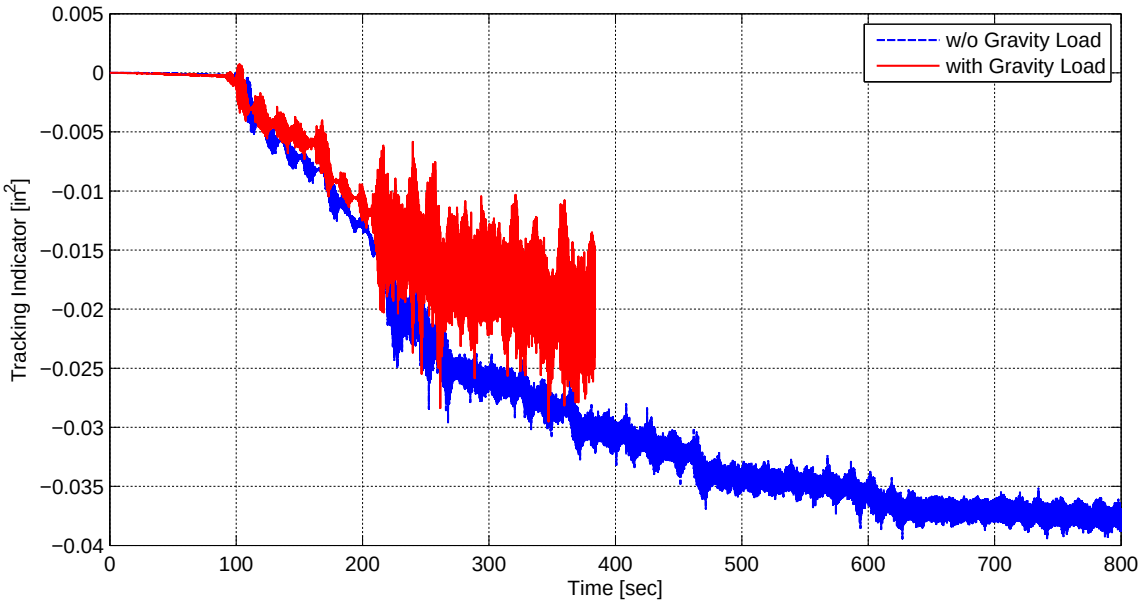


Fig. 6.31 Comparison of tracking performance for test without gravity loads versus test with gravity loads from STS.

Comparison of Analytical Simulation versus Hybrid Simulation:

The numerical model of the μNEES specimen that was developed in Section 6.2.5 is again utilized to perform two purely analytical simulations (AS) of the portal-frame model in OpenSees and compare the numerical results against the test results from the two hybrid simulations (HS). The comparisons of the story-drift time histories as well as the story-shear vs. story-drift hysteresis loops are shown in Fig. 6.32. Overall the analytical simulations were capable of capturing the response of the portal-frame reasonably well, which can be confirmed from the close matches of the story-drift time histories in the first graph.

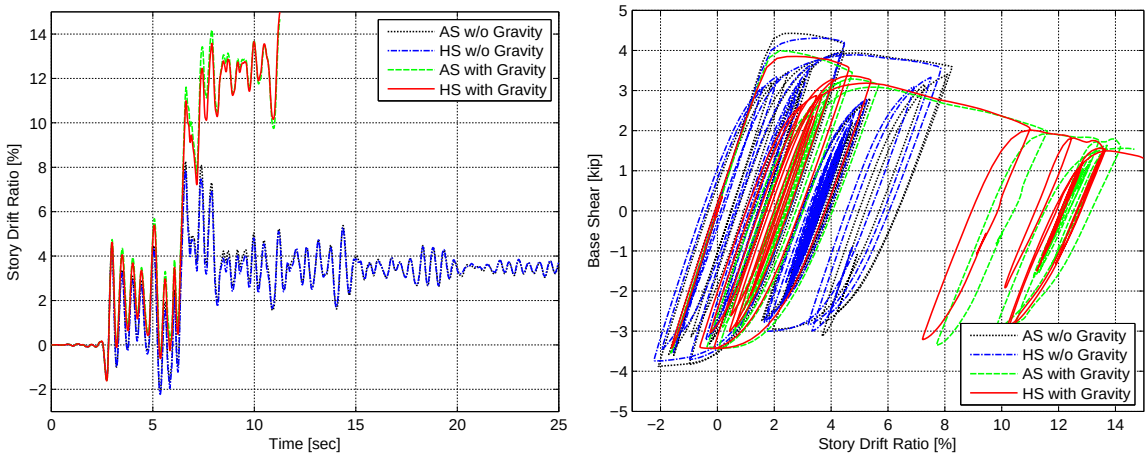


Fig. 6.32 Comparison of story-drift time-histories and total story hysteresis loops of portal-frame model for analytical versus hybrid simulation.

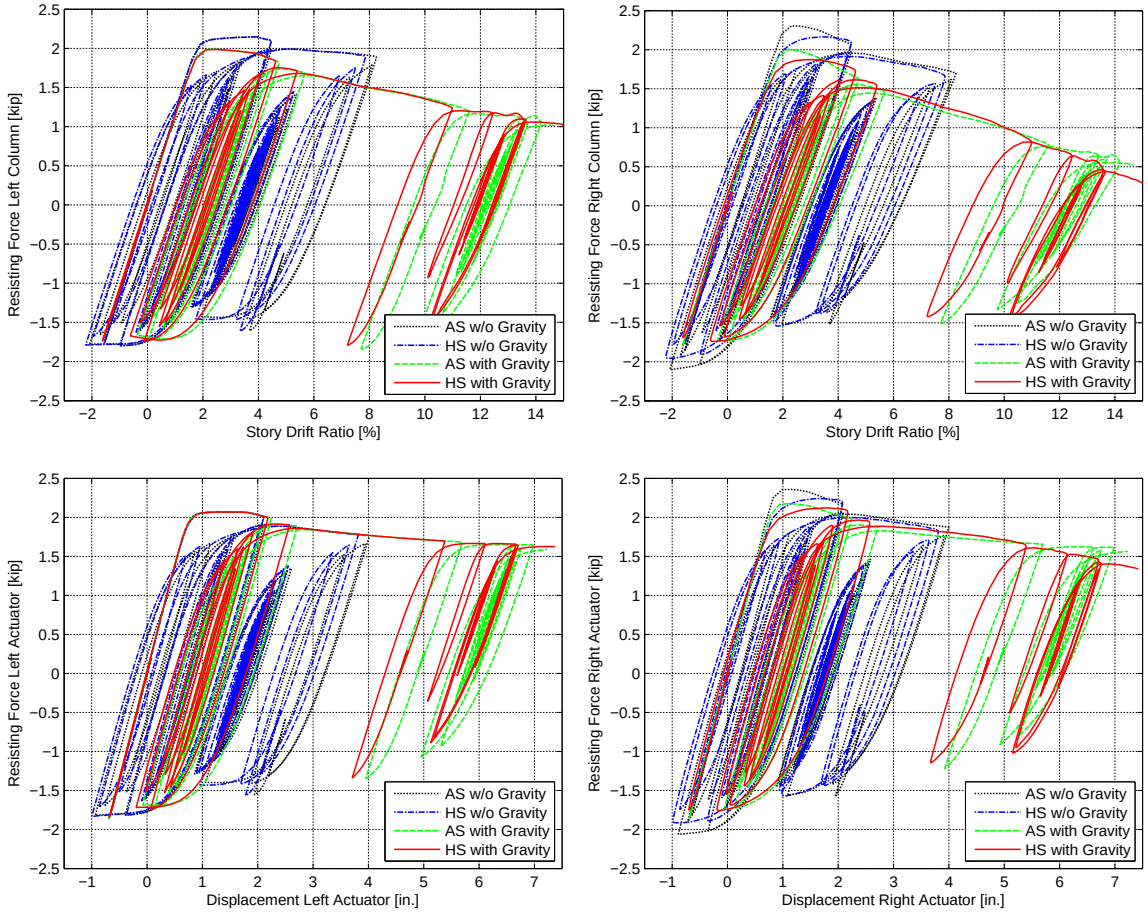


Fig. 6.33 Comparison of element force-story-drift and actuator force-displacement hysteresis loops of portal-frame model for analytical versus hybrid simulation.

However, it can be observed from all the graphs in Fig. 6.32 to Fig. 6.33 that the analytical simulation without gravity loads was clearly more accurate than the one with gravity loads. Furthermore, the divergence between the hybrid and analytical simulations increased with the portal-frame approaching collapse. This can be explained by the complex behavior (severe buckling, necking of the cross-section, crack propagation) of the replaceable steel coupons for large rotations of the clevis (respectively large story-drift-ratios), which was not accurately captured by the simplified finite element model of the specimen.

Nevertheless, as mentioned before, the overall response of the purely analytical and the hybrid structure matched very well, which successfully demonstrated the validity of both models. Particularly, it proved that the OpenFresco software framework, with the novel experimental beam-column element and the other research presented herein, provide a useful and effective set of modules for performing accurate, safe, and cost-effective hybrid simulations of structural collapse.

6.4 SUMMARY

The two novel applications of hybrid simulation, which have been discussed in this chapter, successfully demonstrated and validated some of the features and the overall flexibility of the newly implemented object-oriented hybrid simulation framework presented in Chapter 3. In addition, several other independent developments and implementations from Chapters 4 and 5, such as the integration methods and the event-driven strategies, have been employed to successfully execute two challenging hybrid simulation case studies.

The first application presented a continuous geographically distributed hybrid simulation that was executed in soft real time for the first time. The latest advancements in networking technology (Abilene network) paired with the experimental software framework, OpenFresco, made such real-time geographically distributed hybrid simulation possible. Over the last few years, real-time hybrid simulations have considerably gained significance because many new high-performance structural devices, such as base isolation bearings, energy-dissipation devices and high-performance materials, exhibit intrinsic rate-dependent behavior. Hence, it is of great importance that testing of structural systems incorporating such devices is performed at the correct loading rates even if the hybrid simulations are geographically distributed. The one-bay-

frame structure, the test setup and procedure, the network performance, the transfer system performance and the response of the hybrid model were all discussed and evaluated in detail.

The second case study introduced a novel approach for conducting structural collapse investigations utilizing hybrid simulation. It was demonstrated that the OpenFresco software framework provides an effective set of modules, especially the novel experimental beam-column elements, for performing accurate, safe, and cost-effective hybrid simulations of structural collapse. Furthermore, it was shown that the ability to correctly account for the second-order effects in hybrid models is crucial for simulating collapse of structures under gravity loads. In summary, hybrid simulation of collapse behavior offers three significant advantages over conventional testing methods: (1) the gravity loads and the resulting geometric nonlinearities are represented in the analytical portion of the hybrid model in the computer, eliminating the need for complex active or passive gravity load setups; (2) there is no need to protect expensive test equipment from specimen impact during collapse because the actuator control system will limit the movements of the test specimens; and (3) only the critical, collapse-sensitive, elements of the structure need to be physically tested allowing for a substantial increase in the number of different collapse tests afforded by the same budget. Finally, the portal-frame structure, the test setup and procedure, the transfer system performance and the response of the hybrid model were again discussed and evaluated in detail.

7 Conclusions and Recommendations

7.1 SUMMARY AND CONCLUSIONS

The overarching goal of this research was to explore, develop, and validate advanced techniques to facilitate the wider use of local and geographically distributed hybrid simulation by the research community and permit hybrid simulation to address a range of complex problems of current interest to investigators. To achieve this goal, investigations were conducted to address the following objectives:

1. Review and evaluate existing concepts and frameworks in hybrid simulation and devise and implement an improved object-oriented software framework for experimental testing.
2. Investigate existing time-stepping integration methods, evaluate their applicability to hybrid simulations, and develop and implement improved versions for several methods.
3. Examine existing predictor-corrector algorithms and event-driven strategies for synchronization, and conceive and implement new and improved approaches to synchronize the integration and transfer system processes.

To accomplish these objectives, the research efforts were arranged into the following activities. The first activity was to conduct an extensive literature review to determine the advantages, as well as past and current challenges associated with the hybrid simulation experimental testing technique. Furthermore, the fundamental concept and the different modes of conducting hybrid simulations were discussed. It was concluded that the lack of a common framework for the implementation and deployment of hybrid simulation has posed a major hurdle for those who considered utilizing hybrid simulation. Moreover, it has also limited collaboration among experts in the field. Thus, the second activity was to devise and implement a number of important extensions to an object-oriented software framework for experimental testing (Takahashi and Fenves 2006) to support a large variety of finite element analysis software, structural testing methods, specimen types, testing configurations, control and data-acquisition systems, and network communication protocols. Because in hybrid simulations the step-by-step numerical integration of the semi-discrete equations of motion acts as the computational driver, the third

activity was to examine existing hybrid-simulation specialized integration schemes and develop improved versions for a few of these methods. While the third activity focused on the finite element analysis side of a hybrid simulation, the fourth activity was concerned with investigating methods and strategies for the control and data-acquisition systems on the laboratory side of the testing technique. Specifically, improved event-driven strategies and predictor-corrector algorithms, which are employed to synchronize the integration and transfer system processes, were developed and compared. The fifth and finally activity was to develop and execute a few novel and demonstrative hybrid simulations to validate the software framework, the integration methods, and the event-driven synchronization strategies. A short summary of each activity combined with the key findings is provided next.

7.1.1 Hybrid Simulation Fundamentals

To commence the investigations, it was shown that among the currently well-established experimental testing techniques, hybrid simulation is able to provide a range of important advantages but still faces several important challenges. It was explained that some of the major advantages gained through hybrid simulation are the analytically defined loading, the substructuring capabilities, the ability to test large-scale specimens, the option to utilize affordable testing equipment, the variable speeds of execution, and the possibility to geographically distribute numerical and physical subassemblies. On the other hand, some of the key challenges that still need to be addressed were discussed. These include the implementation of an environment-independent framework for the development and deployment of hybrid simulation, the exploration of force and mixed force/displacement control strategies, dealing with network delays and breakdowns in geographically distributed tests, the development of methods for assessing the goodness of a hybrid simulation, and the execution of hybrid simulations in high-performance computing (HPC) environments.

The four key components, which are required to perform a hybrid simulation, were laid out in a comprehensive way. They consist of a discrete model of the structure including the static and dynamic loading; a transfer system comprising a controller and static or dynamic actuators; the physical specimen that is being tested in the laboratory, including a support against which the actuators of the transfer system can react; and a data-acquisition system including instrumentation devices.

The most common modes of deployment of hybrid simulations, such as local versus geographically distributed tests and slow versus rapid versus real-time tests, were summarized and discussed. A local hybrid simulation was defined by the collocation of all the required components in one testing facility. In contrast, for geographically distributed tests some or all the participating sites are dispersed and need to be connected through wide area networks. Moreover, it was noted that the equations of motion, which are solved by a time-stepping integration method, will by necessity take on slightly different forms depending on the execution speed of the hybrid simulation.

7.1.2 Experimental Software Framework

From the literature review it was found that even though only a small number of basic operations and communication protocols are needed to perform hybrid simulations, hitherto few efforts have been made to implement a universal, environment-independent software framework for developing and deploying systems that are capable of carrying out hybrid simulations. It was explained that a software framework is a reusable design for a system or subsystem and defines the overall architecture of such system, meaning its fundamental components as well as the relationships among them.

Conforming to this definition, an existing software framework for experimental testing (OpenFresco) was redesigned, extended, and implemented to support a large variety of computational drivers, structural testing methods, specimen types, testing configurations, control and data-acquisition systems, and communication protocols. In addition, to accelerate the development and refinement of hybrid simulation, OpenFresco was implemented to be environment independent, robust, transparent, scalable, and easily extensible. It was described how the original development of the software framework is based on the structural analysis approach, in which a hybrid simulation can be viewed as a conventional finite element analysis where physical subassemblies of the structure are embedded in the numerical model. Object-oriented methodologies and a rigorous systems analysis of the operations performed during hybrid simulations were used to determine the fundamental building blocks of the OpenFresco software framework.

Four main software abstractions (classes) had previously been introduced to represent the real-world components required to perform hybrid simulations (Takahashi and Fenves 2006).

These classes are the *ExperimentalElement*, which represents those parts of a structure that are physically tested in a laboratory; the *ExperimentalSite*, which represents the laboratories where the testing takes place or the computational sites where the analysis is carried out; the *ExperimentalSetup,* which represents the possible configurations of the transfer system; and the *ExperimentalControl*, which provides the interface to communicate with a wide variety of control and data-acquisition systems.

One of the key contributions of the research presented here is the application of a multi-tier client/server architecture that was wrapped around the fundamental OpenFresco building blocks. As part of this work, the OpenFresco core components were redesigned and newly implemented before they were combined with the multi-tier client/server architecture. It was shown how the new design supports any finite element analysis software as the computational driver and facilitates a wide variety of local and geographically distributed testing configurations. In addition, these different modes of deployment were explained in detail. Hence, it can be concluded that OpenFresco's new three- and four-tier architectural patterns provide the modularity and flexibility to interface any computational agent running on one or more machines with physical specimens being tested in one or more laboratories.

The final OpenFresco software design that resulted from the object-oriented design phase was summarized utilizing class diagrams, class APIs, and sequence diagrams as are commonly done for software developments. Afterwards, all the concrete classes, which have been implemented as part of the work presented here, and protocols for distributed computation and testing, were discussed. The OpenFresco source-code, documentation, and examples are available at http://openfresco.neesforge.nees.org. Finally, it was shown how OpenSees could be used as the computational framework. It was found that improved performance could be achieved as compared to other finite element analysis software because the two software frameworks can directly be integrated without utilizing network connections.

Overall, it can be concluded that the redesigned, extended, and newly implemented software framework for experimental testing is very flexible, extensible, and capable of interfacing with a wide variety of finite element analysis software and of supporting a large array of testing equipment. Thus, it will be able to boost the development and deployment of hybrid simulation and promote the collaboration among experts in the field.

7.1.3 Integration Methods

After the OpenFresco middleware, which allows any finite element analysis software to be connected to a laboratory, had successfully been redesigned, the goal was to investigate another of the key components required to perform hybrid simulations; that is, the time-stepping integration methods, which act as the computational drivers and need to be provided by or implemented in the finite element analysis software.

A wide range of integration schemes from both the class of direct integration methods and the class of Runge-Kutta methods were investigated and evaluated in terms of their applicability to hybrid simulation. The first step in these investigations identified and summarized special integration scheme properties required for the reliable, accurate and fast execution of hybrid simulations. It was concluded that explicit noniterative methods are well suited for hybrid simulation and easy to implement, as long as it is possible to satisfy their stability limits with a reasonable time-step size. In contrast, it was demonstrated that unconditionally stable iterative implicit methods should be used for hybrid simulation under only one of two conditions: either a fixed user-specified number of iterations is utilized in the equilibrium solution algorithm or the event-driven algorithms that were introduced in another part of this report need to automatically adjust the time intervals over which displacement increments are imposed. For the former approach, improved versions of existing algorithms were developed that achieve superior convergence in the iteration process. Third, it was shown that the operator-splitting methods and Rosenbrock methods, which are unconditionally stable, relatively easy to implement, and computationally almost as efficient as explicit methods, are excellent for solving the equations of motion during hybrid simulations. For the operator-splitting schemes, two new methods were proposed, the generalized-alpha-OS and the Modified generalized-alpha-OS method. They were described in detail and it was concluded that as long as the hybrid model does not exhibit severe geometric nonlinearities, the two methods are very attractive integration schemes for hybrid simulation. Another key component of this work is the thorough investigation of the applicability of Runge-Kutta methods to hybrid simulation.

The accuracy of both the class of direct integration methods and the class of Runge-Kutta methods considered was also evaluated in terms of two measures, numerical dissipation and dispersion. It was concluded that for hybrid simulation the integration schemes should preferably provide some user-adjustable amount of algorithmic energy dissipation in high-frequency modes

to damp out any spurious oscillations that might get excited by experimental errors. In contrast, it was concluded that the numerical dispersion, meaning the relative period elongation, should be as small as possible for maximal accuracy of the integration method.

Finally, all the integration methods introduced were summarized in form of pseudo-code to assist with their implementation into finite element software. In addition, most of the methods have already been implemented in OpenSees.

7.1.4 Event-Driven Strategies

The predictor-corrector algorithms that provide the synchronization of the integration and transfer system processes and need to be implemented on the control and data-acquisition system side in a real-time environment were investigated next. Previous developments, such as methods for continuous and real-time testing as well as the three-loop hardware architecture (Mosqueda 2003) that makes continuous geographically distributed testing possible, were discussed first.

Several improved predictor-corrector algorithms, which incorporate the last predicted displacement command into the corrector formulas, were then introduced. It was shown that these algorithms achieve improved performance in the sense that they generate continuous displacement command signals without inconsistencies in the switching interval. Furthermore, two new predictor-corrector algorithms that are based on velocities and accelerations, in addition to displacements, were proposed and shown to achieve improved accuracy while preserving continuity in the generated displacement command signals. The theories of all synchronization algorithms introduced were covered in detail, and their performance and accuracy evaluated graphically in terms of amplitude change and a phase shift. Hence, it can be concluded that the new and improved synchronization algorithms, which have been implemented in the C programming language and are available to users, provide means for performing more accurate continuous hybrid simulations and should therefore be utilized in place of the original algorithms.

The event-driven solution strategies, which employ the previously described algorithms for prediction and correction, were investigated next. Again, the existing strategy was evaluated first, and based on the deficiencies identified, new and improved solution strategies were suggested. Most of the new strategies are based on adaptive control concepts that automatically adjust parameters of the event-driven controller, such as time steps, actuator velocities etc.,

according to some suitable performance criteria. It was shown that one of the improved strategies successfully generates continuous velocity transition (whenever test execution needs to be slowed down or sped up) instead of the discontinuous ones of the original strategy. Another one of the new solution strategies was specifically developed to work in conjunction with the implicit Newmark direct integration method (that would otherwise be inappropriate for hybrid simulation), by adapting time intervals over which displacement commands are imposed. Finally, except for the adaptive simulation time-step strategies and the adaptive integration time-step strategies, which still need to be tested and evaluated in a real physical environment, all the other event-driven strategies have been implemented in Simulink/Stateflow (Mathworks) and are available to users for performing hybrid simulations.

As a result, it can be concluded that the new and improved predictor-corrector algorithms and event-driven strategies provide an important and useful set of tools to perform continuous hybrid simulations in which consistent and more accurate commands are generated for the control systems.

7.1.5 Novel Hybrid Simulation Examples

To confirm the robustness, flexibility, and usability of the extended OpenFresco software framework and to validate the integration methods and the event-driven synchronization strategies introduced, two new, interesting, and challenging hybrid simulation examples were carried out.

The first application presented a continuous geographically distributed hybrid simulation that was executed in soft real time for the first time in history. It was shown that OpenFresco's newly implemented Shadow and Actor experimental sites combined with the latest advancements in networking technology (Abilene network) made such real-time geographically distributed hybrid simulation possible. Furthermore, it can be concluded that the improved predictor-corrector algorithms and event-driven strategies successfully guaranteed the continuous execution of the test, as well as the smooth slow downs and speed ups of the experiment in case of network delays.

It was discussed that over the last few years, real-time hybrid simulations have gained considerably in significance because the many new high-performance structural devices, such as base isolation bearings, energy-dissipation devices, and high-performance materials, exhibit

intrinsic rate-dependent behavior. Thus, it was concluded that it is of great importance that testing of structural systems incorporating such devices is performed at the correct loading rates even if the hybrid simulations are geographically distributed. It was successfully demonstrated that the redesigned, extended and newly implemented software framework for experimental testing provides the means to perform such hybrid simulations.

The second application introduced a novel approach for conducting structural collapse investigations utilizing hybrid simulation. It was shown that the OpenFresco software framework provides an effective set of modules, especially the novel experimental beam-column elements, for performing accurate, safe and cost-effective hybrid simulations of structural collapse. Furthermore, it was concluded that the modified implicit Newmark integration scheme with 10 sub-steps is capable of generating an accurate response of the portal-frame despite the significant geometric nonlinearities.

Finally, it was concluded that hybrid simulation of structural collapse offers three important advantages over conventional testing methods: (1) the gravity loads and the resulting geometric nonlinearities are represented in the analytical portion of the hybrid model in the computer, eliminating the need for complex active or passive gravity load setups; (2) there is no need to protect expensive test equipment from specimen impact during collapse because the actuator control system will limit the movements of the test specimens; and (3) only the critical, collapse-sensitive elements of the structure need to be physically tested allowing for a substantial increase in the number of different collapse tests afforded by the same budget.

7.2 RECOMMENDATIONS FOR FUTURE WORK

The topics cover by this research constitute a valuable contribution that provides the research community with an extensive set of tools, such as a robust and flexible software framework and a comprehensive compilation of time-stepping integration techniques and synchronization strategies, to perform advanced hybrid simulations to experimentally investigate the behavior of complex structures. In the process of developing and implementing many of the discussed topics several research areas were identified which could benefit from further study.

OpenFresco Software Framework:

1. Enhance the ability of the hybrid simulation software framework to simulate in dry-run mode test specimens and transfer systems, including controllers, actuators, hydraulic supply systems, and instrumentation. Such capabilities to conduct dry runs of an experiment before the actual test are critical in determining control parameters, and in confirming that various analysis parameters are input properly ensuring that the hybrid simulation can be run safely. The OpenFresco software framework currently permits simulation of the specimen in dry-run mode, but features are not adequate to simulate the full system including critical aspects of controllers, hydraulic control systems, etc., necessary to assess overall stability and safety.
2. Improve the support for rapid and soft real-time geographically distributed hybrid simulations and verify that the OpenFresco software framework provides the necessary capabilities for hard real-time hybrid simulations. This is a rapidly growing research area with several NEES equipment sites and individual research projects undertaking projects that require real-time or rapid test execution. The following activities should be included: Implement methods into the OpenFresco software framework for compensating for inertial and viscous damping forces that are being generated in the physically tested portions of a structure during the rapid or real-time execution of a hybrid simulation (see Chapter 2). Extend the OpenFresco software framework so that it can be employed to execute smart shaking table tests by performing a hybrid simulation where a simulator platform is controlled in real time. Investigate methods to further improve the speed and reliability of geographically distributed hybrid simulations and to take full advantage of the capabilities of the next generation Internet.
3. Several NEES equipment sites and researchers are currently working towards the ability to conduct force and mixed force/displacement controlled hybrid simulations. Thus, it is important to collaborate with these equipment sites and researchers to make any changes needed to the OpenFresco software framework to enable force and mixed force/displacement control, as well as switching during simulation between force and displacement control.
4. One of the main features of the OpenFresco software framework is its ability to allow users to employ any finite element analysis software of their choice. Currently, the

implementations with OpenSees, Matlab, LS-Dyna, and SimCor work well. However, additional interfaces for other widely used finite element software packages, such as Abaqus and SAP2000, should be investigated and interfaces be implemented.

5. One of the NEES objectives is to foster the open exchange of data and information among researchers and practicing engineers through shared, web-accessible repositories. Hence, it is essential to implement recording capabilities for all the objects in the OpenFresco software framework and investigate approaches for automatically uploading all the recorded data into the NEES data repository.

Integration Methods:

1. Investigate the changes or additions that are required in OpenSees in order to implement all the discussed Runge-Kutta integration methods, especially the real-time compatible Rosenbrock methods. Because for Runge-Kutta methods the second-order differential equation is converted into a system of first-order ordinary differential equations, the size of the linear system of equations that needs to be solved during the analysis doubles. This will entail some new additions to OpenSees to assemble and solve these distinctive linear systems of equations.
2. The application of mixed time implicit-explicit integration schemes (Liu and Belytschko 1982) to hybrid simulation should be investigated. Contrary to implicit-explicit integration methods (Hughes and Liu 1978), which were employed for hybrid simulation by Dermitzakis and Mahin (1985), the partition procedures by Liu and Belytschko (1982) permit different time-integration algorithms with different time steps to be used simultaneously in different parts of the finite element mesh.

Event-Driven Strategies:

1. Many simulation results but only a limited number of actual tests were performed to validate the newly introduced predictor-corrector algorithms. Hence, additional real hybrid simulations should be performed to enforce the validation of the algorithms under a broader range of conditions. Specifically, the algorithms that are based on velocities and accelerations should be further investigated if the velocities and accelerations are provided by the integration methods (especially the Runge-Kutta methods once they are implemented in OpenSees).

2. To make the adaptive simulation and integration time-step strategies available to hybrid simulation users and researchers, they should be implemented in Simulink/Stateflow (Mathworks) so that they can be validated through actual tests in a real-time environment prior to releasing them.

Applications:

1. There is a range of hybrid simulation applications that are challenging to execute because of the following known issues. It is difficult to control three-dimensional systems utilizing an array of actuators due to their kinematic interaction. Furthermore, controlling experimental subassemblies with large physical masses is very challenging because of inertial force interactions. This becomes even more difficult if high loading rates are necessary due to the velocity dependence of structural devices. As a result, example applications should be designed to perform hybrid simulations to investigate these issues and research means for dealing with these difficulties.

REFERENCES

Ahmadizadeh, M., et al. [2008]. "Compensation of actuator delay and dynamics for real-time hybrid structural simulation." Earthquake Engineering & Structural Dynamics 37(1): 21-42.

Bathe, K.-J. [1996]. Finite element procedures. Englewood Cliffs, NJ, Prentice Hall.

Bayer, V., et al. [2005]. "On real-time pseudo-dynamic sub-structure testing: algorithm, numerical and experimental results." Aerospace Science and Technology 9(3): 223-232.

Belytschko, T. and Hughes, T. J. R. [1983]. Computational methods for transient analysis. Amsterdam; New York, NY, North-Holland.

Blakeborough, A., et al. [2001]. "The development of real-time substructure testing." Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences 359(1786): 1869-1891.

Bonelli, A. and Bursi, O. S. [2004]. "Generalized-alpha methods for seismic structural testing." Earthquake Engineering & Structural Dynamics 33(10): 1067-1102.

Bonnet, P. A., et al. [2008]. "Evaluation of numerical time-integration schemes for real-time hybrid testing." Earthquake Engineering & Structural Dynamics 37(13): 1467-1490.

Booch, G., et al. [2005]. The unified modeling language user guide. Upper Saddle River, NJ, Addison-Wesley.

Burden, R. L. and Faires, J. D. [2001]. Numerical analysis. Pacific Grove, CA, Brooks/Cole-Thomson Learning.

Bursi, O. S., et al. [2008]. "Novel coupling Rosenbrock-based algorithms for real-time dynamic substructure testing." Earthquake Engineering & Structural Dynamics 37(3): 339-360.

Bursi, O. S., et al. [1994]. "Pseudodynamic testing of strain-softening systems with adaptive time steps." Earthquake Engineering & Structural Dynamics 23(7): 745-760.

Bursi, O. S. and Shing, P. S. B. [1996]. "Evaluation of some implicit time-stepping algorithms for pseudodynamic tests." Earthquake Engineering & Structural Dynamics 25(4): 333-355.

Butcher, J. C. [2003]. Numerical methods for ordinary differential equations. Chichester, West Sussex, England ; Hoboken, NJ, J. Wiley.

Campbell, S. and Stojadinovic, B. [1998]. A system for simultaneous pseudodynamic testing of multiple substructures. Proceedings, Sixth U.S. National Conference on Earthquake Engineering. Seattle, WA.

Carnegie Mellon SEI. [2008]. "Software technology roadmap." from http://www.sei.cmu.edu/str.

Chang, S.-Y. [2002]. "Explicit pseudodynamic algorithm with unconditional stability." Journal of Engineering Mechanics 128(9): 935-947.

Chang, S. Y. and Sung, Y. C. [2006]. An enhanced explicit pseudodynamic algorithm with unconditional stability. Proceedings, Eighth U.S. National Conference on Earthquake Engineering. San Francisco, CA.

Chopra, A. K. [2001]. Dynamics of structures: theory and applications to earthquake engineering. Upper Saddle River, NJ, Prentice Hall.

Chung, J. and Hulbert, G. M. [1993]. "A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-alpha method." Journal of Applied Mechanics, ASME 60(2): 371-375.

Combescure, D. and Pegon, P. [1997]. "Alpha-operator splitting time integration technique for pseudodynamic testing error propagation analysis." Soil Dynamics and Earthquake Engineering 16(7-8): 427-443.

Cook, R. D. [2002]. Concepts and applications of finite element analysis. New York, NY, Wiley.

Darby, A. P., et al. [1999]. "Real-time substructure tests using hydraulic actuator." Journal of Engineering Mechanics 125(10): 1133-1139.

Darby, A. P., et al. [2001]. "Improved control algorithm for real-time substructure testing." Earthquake Engineering & Structural Dynamics 30(3): 431-448.

de Souza, R. M. [2000]. Force-based finite element for large displacement inelastic analysis of frames, University of California, Berkeley. Ph.D.: 205.

Deitel, H. M. and Deitel, P. J. [2003]. C++ how to program. Upper Saddle River, NJ, Prentice Hall.

Dermitzakis, S. N. and Mahin, S. A. [1985]. Development of substructuring techniques for on-line computer controlled seismic performance testing. Berkeley, CA, Earthquake Engineering Research Center: 153 pages.

Dorka, U. E. [2002]. Hybrid experimental - numerical simulation of vibrating structures. International Conference WAVE2002. Okayama, Japan.

Dorka, U. E. and Heiland, D. [1991]. Fast online earthquake utilizing a novel PC supported measurement and control concept. 4th Conference on Structural Dynamics. Southampton, UK.

dSpace Inc. [2008]. "dSPACE." from http://www.dspaceinc.com/ww/en/inc/home.cfm.

Elkhoraibi, T. and Mosalam, K. M. [2007]. "Towards error-free hybrid simulation using mixed variables." Earthquake Engineering & Structural Dynamics 36(11): 1497-1522.

Erlicher, S., et al. [2002]. "The analysis of the generalized-α method for non-linear dynamic problems." Computational Mechanics 28(2): 83-104.

Fenves, G. L. [1990]. "Object-oriented programming for engineering software development." Engineering with Computers 6(1): 1-15.

Fenves, G. L., et al. [2004]. An object-oriented software environment for collaborative network simulation. Proceedings, 13th World Conference on Earthquake Engineering. Vancouver, BC, Canada.

Filippou, F. C. and Fenves, G. L. [2004]. Methods of analysis for earthquake-resistant structures. Earthquake Engineering: From Engineering Seismology to Performance-Based Engineering. Boca Raton, FL, CRC Press.

Gamma, E., et al. [1995]. Design patterns: elements of reusable object-oriented software. Reading, MA, Addison-Wesley.

Gawthrop, P. J., et al. [2007]. "Robust real-time substructuring techniques for under-damped systems." Structural Control and Health Monitoring 14(4): 591-608.

Gourley, D. and Totty, B. [2002]. Http: the definitive guide. Beijing; Cambridge, MA, O'Reilly.

Hairer, E., et al. [2000]. Solving ordinary differential equations I, nonstiff problems. Berlin; New York, NY, Springer-Verlag.

Hairer, E. and Wanner, G. [2002]. Solving ordinary differential equations II, stiff and differential-algebraic problems. Berlin; New York, NY, Springer-Verlag.

Harris, H. G., et al. [1999]. Structural modeling and experimental techniques. Boca Raton, FL, CRC Press.

Heun, K. [1900]. "Neue Methoden zur approximativen Integration der Differentialgleichungen einer unabhängigen Veränderlichen." Zeitschrift für Mathematik und Physik 45: 23-38.

Hilber, H. M. and Hughes, T. J. R. [1978]. "Collocation, dissipation and 'overshoot' for time integration schemes in structural dynamics." Earthquake Engineering and Structural Dynamics 6(1): 99-117.

Hilber, H. M., et al. [1977]. "Improved numerical dissipation for time integration algorithms in structural dynamics." Earthquake Engineering and Structural Dynamics 5(3): 283-292.

Horiuchi, T., et al. [2000]. Development of a real-time hybrid experimental system using a shaking table. Proceedings, 12th World Conference on Earthquake Engineering. Auckland, New Zealand.

Horiuchi, T., et al. [1999]. "Real-time hybrid experimental system with actuator delay compensation and its application to a piping system with energy absorber." Earthquake Engineering & Structural Dynamics 28(10): 1121-1141.

Horiuchi, T. and Konno, T. [2001]. "A new method for compensating actuator delay in real-time hybrid experiments." Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences 359(1786): 1893-1909.

Horiuchi, T., et al. [1996]. Development of a real-time hybrid experimental system with actuator delay compensation. Proceedings, 11th World Conference on Earthquake Engineering. Acapulco, México.

Hubbard, P., et al. [2004]. Protocol specification for the LabView NTCP plugin. San Diego, CA, NEESit Technical Report.

Hughes, T. J. R. [2000]. The finite element method: linear static and dynamic finite element analysis. Mineola, NY, Dover Publications.

Hughes, T. J. R. and Liu, W. K. [1978]. "Implicit-explicit finite elements in transient analysis: implementation and numerical examples." Journal of Applied Mechanics, ASME 45(2): 375-378.

Hughes, T. J. R. and Liu, W. K. [1978]. "Implicit-explicit finite elements in transient analysis: stability theory." Journal of Applied Mechanics, ASME 45(2): 371-374.

Hughes, T. J. R., et al. [1979]. "Implicit-explicit finite elements in nonlinear transient analysis " Computer Methods in Applied Mechanics and Engineering 17(18): 159-182.

Hulbert, G. M. and Chung, J. [1996]. "Explicit time integration algorithms for structural dynamics with optimal numerical dissipation." Computer Methods in Applied Mechanics and Engineering 137(2): 175-188.

Humar, J. L. [2002]. Dynamics of structures. Exton, PA, A.A. Balkema Publishers.

Igarashi, A., et al. [1993]. Development of the pseudodynamic technique for testing a full scale 5-story shear wall structure. U.S. - Japan Seminar, Development and Future Dimensions of Structural Testing Techniques. Honolulu, HI.

Jung, R.-Y. and Shing, P. B. [2006]. "Performance evaluation of a real-time pseudodynamic test system." Earthquake Engineering & Structural Dynamics 35(7): 789-810.

Jung, R.-Y., et al. [2007]. "Performance of a real-time pseudodynamic test system considering nonlinear structural response." Earthquake Engineering & Structural Dynamics 36(12): 1785-1809.

Kaps, P. and Wanner, G. [1981]. "A study of Rosenbrock-type methods of high order." Numerische Mathematik 38(2): 279-298.

Kuhl, D. and Crisfield, M. A. [1999]. "Energy-conserving and decaying algorithms in non-linear structural dynamics." International Journal for Numerical Methods in Engineering 45(5): 569-599.

Kutta, W. [1901]. "Beitrag zur näherungsweisen Integration totaler Differentialgleichungen." Zeitschrift für Mathematik und Physik 46(435-453).

Kwon, O.-S., et al. [2005]. "A framework for multi-site distributed simulation and application to complex structural systems." Journal of Earthquake Engineering 9(5): 741-753.

Lambert, J. D. [1991]. Numerical methods for ordinary differential systems: the initial value problem. Chichester ; New York, NY, Wiley.

Liu, W. K. and Belytschko, T. [1982]. "Mixed-time implicit-explicit finite elements for transient analysis." Computers and Structures 15(4): 445-450.

Liu, W. K., et al. [1984]. "Implementation and accuracy of mixed-time implicit-explicit methods for structural dynamics." Computers & Structures 19(4): 521-530.

Livermore Software Technology Corp. [2008]. "LS-DYNA." from http://www.lstc.com.

Magonette, G. [2001]. "Development and application of large-scale continuous pseudo-dynamic testing techniques." Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences 359(1786): 1771-1799.

Magonette, G. E. and Negro, P. [1998]. "Verification of the pseudodynamic test method." European Earthquake Engineering XII(1): 40-50.

Mahin, S. A., et al. [1989]. "Pseudodynamic test method - current status and future directions." Journal of Structural Engineering 115(8): 2113-2128.

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

Mahin, S. A. and Williams, M. E. [1980]. Computer controlled seismic performance testing. Dynamic Response of Structures: Experimentation, Observation, Prediction and Control, American Society of Civil Engineers, New York, NY.

Manring, N. [2005]. Hydraulic control systems. Hoboken, NJ, John Wiley & Sons.

Maplesoft. [2008]. "Maple." from http://www.maplesoft.com.

Mathworks. [2008]. "MATLAB, Simulink, Stateflow, xPC Target." from http://www.mathworks.com.

McKenna, F. T. [1997]. Object-oriented finite element programming: frameworks for analysis, algorithms and parallel computing, University of California, Berkeley. Ph.D.: 247.

Mercan, O. and Ricles, J. M. [2007]. "Stability and accuracy analysis of outer loop dynamics in real-time pseudodynamic testing of SDOF systems." Earthquake Engineering & Structural Dynamics 36(11): 1523-1543.

Morgan, T. A. [2007]. The use of innovative base isolation systems to achieve complex seismic performance objectives, University of California, Berkeley. Ph.D.: 323.

Mosqueda, G. [2003]. Continuous hybrid simulation with geographically distributed substructures, University of California, Berkeley. Ph.D.: 232.

Mosqueda, G., et al. [2005]. Implementation and accuracy of continuous hybrid simulation with geographically distributed substructures. Berkeley, CA, Earthquake Engineering Research Center.

Mosqueda, G., et al. [2007]. "Real-time error monitoring for hybrid simulation. Part II: Structural response modification due to errors." Journal of Structural Engineering (8): 1109-1117.

Mosqueda, G., et al. [2007]. "Real-time error monitoring for hybrid simulation. Part I: methodology and experimental verification." Journal of Structural Engineering (8): 1100-1108.

Mosqueda, G., et al. [2004]. Experimental and analytical studies of the friction pendulum system for the seismic protection of simple bridges. Berkeley, CA, Earthquake Engineering Research Center.

MTS. [2008]. "Civil engineering testing solutions." from http://www.mts.com.

Nakashima, M. [1985]. "Part 2: Relationship between integration time interval and accuracy of displacement, velocity, and acceleration responses in pseudo dynamic testing." Journal of Structural and Construction Engineering (Transactions of AIJ) (358): 35-42.

Nakashima, M. [1985]. "Part 1: Relationship between integration time interval and response stability in pseudo dynamic testing (stability and accuracy behavior of pseudo dynamic response)." Journal of Structural and Construction Engineering (Transactions of AIJ) (353): 29-36.

Nakashima, M. [2001]. "Development, potential, and limitations of real-time online (pseudo-dynamic) testing." Philosophical Transactions of the Royal Society: Mathematical, Physical and Engineering Sciences 359(1786): 1851-1867.

Nakashima, M., et al. [1988]. Feasibility of pseudo dynamic test using substructuring techniques. Proceedings, Ninth World Conference on Earthquake Engineering. Tokyo, Japan.

Nakashima, M., et al. [1990]. Integration techniques for substructure pseudo dynamic test. Proceedings, Fourth U.S. National Conference on Earthquake Engineering. Palm Springs, CA.

Nakashima, M. and Kato, H. [1987]. Experimental error growth behavior and error growth control in on-line computer test control method. Building Research Institute, Ministry of Construction, Tsukuba, Japan: 145 pages.

Nakashima, M. and Kato, H. [1988]. "Part 3: experimental error growth in pseudo dynamic testing (stability and accuracy behavior of pseudo dynamic response)." Journal of Structural and Construction Engineering (Transactions of AIJ) (386): 36-48.

Nakashima, M. and Kato, H. [1989]. "Part 4: Control of experimental error growth in pseudo dynamic testing--Stability and accuracy behavior of pseudo dynamic response." Journal of Structural and Construction Engineering (Transactions of AIJ) (401): 129-138.

Nakashima, M., et al. [1992]. "Development of real-time pseudo dynamic testing." Earthquake Engineering & Structural Dynamics 21(1): 79-92.

Nakashima, M. and Masaoka, N. [1999]. "Real-time on-line test for MDOF systems." Earthquake Engineering & Structural Dynamics 28(4): 393-420.

Newmark, N. M. [1959]. "A method of computation for structural dynamics." Journal of Engineering Mechanics, ASCE 67.

Nise, N. S. [2004]. Control systems engineering. Hoboken, NJ, Wiley.

OpenFresco. [2008]. "Open Framework for Experimental Setup and Control." from http://openfresco.neesforge.nees.org.

OpenSees. [2008]. "Open System for Earthquake Engineering Simulation." from http://opensees.berkeley.edu.

Pan, P., et al. [2005]. "Online test using displacement-force mixed control." Earthquake Engineering & Structural Dynamics 34(8): 869-888.

Pan, P., et al. [2005]. "Online hybrid test by internet linkage of distributed test-analysis domains." Earthquake Engineering & Structural Dynamics 34(11): 1407-1425.

Pan, P., et al. [2006]. "Development of peer-to-peer (P2P) internet online hybrid test system." Earthquake Engineering & Structural Dynamics 35(7): 867-890.

Pinto, A. V., et al. [2004]. "Pseudo-dynamic testing of bridges using non-linear substructuring." Earthquake Engineering & Structural Dynamics 33(11): 1125-1146.

Plesha, M. E. and Belytschko, T. [1985]. "A constitutive operator splitting method for nonlinear transient analysis." Computers & Structures 20(4): 767-777.

Rodgers, J. E. and Mahin, S. A. [2004]. Effects of connection hysteretic degradation on the seismic behavior of steel moment-resisting frames. Berkeley, CA, Pacific Earthquake Engineering Research Center.

Runge, C. [1895]. "Ueber die numerische Auflösung von Differentialgleichungen." Mathematische Annalen 46: 167-178.

Sandee, J. H., et al. [2005]. Event-driven control as an opportunity in the multidisciplinary development of embedded controllers, Technische Universiteit Eindhoven, Dept. of Electrical Engineering, Control Systems Group.

Schellenberg, A. and Mahin, S. [2006]. Integration of hybrid simulation within the general-purpose computational framework Opensees. Proceedings, Eighth U.S. National Conference on Earthquake Engineering. San Francisco, CA.

Schellenberg, A., et al. [2007]. A software framework for hybrid simulation of large structural systems. Proceedings, 2007 Structures Congress. Long Beach, CA.

Schneider, S. P. and Roeder, C. W. [1994]. "An inelastic substructure technique for the pseudodynamic test method." Earthquake Engineering & Structural Dynamics 23(7): 761-775.

Scott, M. H. and Fenves, G. L. [2003]. A Krylov subspace accelerated Newton algorithm. ASCE Structures Congress. Seattle, WA.

Shampine, L. F. [1982]. "Implementation of Rosenbrock methods." ACM Transactions on Mathematical Software 8(2): 93-113.

Shing, P. B., et al. [1994]. "Pseudodynamic tests of a concentrically braced frame using substructuring techniques." Journal of Constructional Steel Research 29(1-3): 121-148.

Shing, P. B. and Mahin, S. A. [1983]. Experimental error propagation in pseudodynamic testing. Berkeley, CA, Earthquake Engineering Research Center: 175 pages.

Shing, P. B., et al. [2002]. Conceptual design of fast hybrid test system at the University of Colorado. Proceedings, Seventh U.S. National Conference on Earthquake Engineering. Boston, MA.

Shing, P. B., et al. [2004]. Nees fast hybrid test system at the University of Colorado. Proceedings, 13th World Conference on Earthquake Engineering. Vancouver, BC, Canada.

Shing, P. S. B. and Mahin, S. A. [1984]. Pseudodynamic test method for seismic performance evaluation: theory and implementation. Berkeley, CA, Earthquake Engineering Research Center: 162 pages.

Shing, P. S. B. and Mahin, S. A. [1987]. "Cumulative experimental errors in pseudodynamic tests." Earthquake Engineering & Structural Dynamics 15(4): 409-424.

Shing, P. S. B. and Mahin, S. A. [1987]. "Elimination of spurious higher-mode response in pseudodynamic tests." Earthquake Engineering & Structural Dynamics 15(4): 425-445.

Shing, P. S. B., et al. [1996]. "Application of pseudodynamic test method to structural research." Earthquake Spectra 12(1): 29-56.

Shing, P. S. B. and Vannan, M. T. [1991]. "Implicit time integration for pseudodynamic tests: convergence and energy dissipation." Earthquake Engineering & Structural Dynamics 20(9): 809-819.

Shing, P. S. B., et al. [1991]. "Implicit time integration for pseudodynamic tests." Earthquake Engineering & Structural Dynamics 20(6): 551-576.

Simo, J. C. and Hughes, T. J. R. [1998]. Computational inelasticity. New York, NY, Springer.

Sivaselvan, M. V. [2006]. A unified view of hybrid seismic simulation algorithms. Boulder, CO, NEES at CU-Boulder.

Smith, S. W. [1997]. The scientist and engineer's guide to digital signal processing. San Diego, CA, California Technical Pub.

Systran. [2008]. "SCRAMNet+ network." from http://www.cwcembedded.com.

Tada, M. and Pan, P. [2007]. "A modified operator splitting (OS) method for collaborative structural analysis (CSA)." International Journal for Numerical Methods in Engineering 72(4): 379-396.

Takahashi, Y. and Fenves, G. L. [2006]. "Software framework for distributed experimental–computational simulation of structural systems." Earthquake Engineering & Structural Dynamics 35(3): 267-291.

Takanashi, K. and Nakashima, M. [1987]. "Japanese activities on on-line testing." Journal of Engineering Mechanics 113(7): 1014-1032.

Takanashi, K., et al. [1975]. "Non-linear earthquake response analysis of structures by a computer-actuator online system--Part 1: detail of the system." Transactions of the Architectural Institute of Japan (229): 77-83.

Tcl Developer Xchange. [2008]. "Tcl/Tk." from http://www.tcl.tk.

Tedesco, J. W., et al. [1999]. Structural dynamics: theory and applications. Menlo Park, CA, Addison-Wesley.

Thewalt, C. R. and Mahin, S. A. [1987]. Hybrid solution techniques for generalized pseudodynamic testing. Berkeley, CA, Earthquake Engineering Research Center: 142 pages.

Thewalt, C. R. and Mahin, S. A. [1995]. "An unconditionally stable hybrid pseudodynamic algorithm." Earthquake Engineering & Structural Dynamics 24(5): 723-731.

Thewalt, C. R. and Roman, M. [1994]. "Performance parameters for pseudodynamic tests." Journal of Structural Engineering 120(9): 2768-2781.

Thiele, K. [2000]. Pseudodynamische Versuche an Tragwerken mit grossen Steifigkeitsanderungen und mehreren Freiheitsgraden. Zurich, Switzerland, Federal Institute of Technology. Ph.D.: 168.

Timoshenko, S. [1961]. Theory of elastic stability. New York, NY, McGraw-Hill.

Uriz, P. [2005]. Towards earthquake resistant design of concentrically braced steel structures, University of California, Berkeley. Ph.D.: 436.

US NSF. [2008]. "George E. Brown Jr. Network for Earthquake Engineering Simulation (NEES)." from http://www.nees.org.

Verwer, J. G. [1981]. "An analysis of Rosenbrock methods for nonlinear stiff initial value problems." SIAM Journal on Numerical Analysis 19(1): 155-170.

Wallace, M. I., et al. [2005]. "Stability analysis of real-time dynamic substructuring using delay differential equation models." Earthquake Engineering & Structural Dynamics 34(15): 1817-1832.

Wallace, M. I., et al. [2005]. "An adaptive polynomial based forward prediction algorithm for multi-actuator real-time dynamic substructuring." Proceedings of the Royal Society of London A 461(2064): 3807-3826.

Wang, T., et al. [2006]. "On-line hybrid test combining with general-purpose finite element software." Earthquake Engineering & Structural Dynamics 35(12): 1471-1488.

Wang, Y.-P., et al. [2001]. "Modified state-space procedures for pseudodynamic testing." Earthquake Engineering & Structural Dynamics 30(1): 59-80.

Wolfram. [2008]. "Mathematica." from http://www.wolfram.com.

Wu, B., et al. [2007]. "Equivalent force control method for generalized real-time substructure testing with implicit integration." Earthquake Engineering & Structural Dynamics 36(9): 1127-1149.

Yang, T. Y. [2006]. Performance evaluation of innovative steel braced frames, University of California, Berkeley. Ph.D.: 245.

Zayas, V. A., et al. [1989]. Feasibility and performance studies on improving the earthquake resistance of new and existing buildings using the friction pendulum system. Berkeley, CA, Earthquake Engineering Research Center: 302 pages.

Zayas, V. A., et al. [1987]. The FPS earthquake resisting system: experimental report. Berkeley, CA, Earthquake Engineering Research Center: 98 pages.

Zhong, W. [2005]. Fast hybrid test system for substructure evaluation, University of Colorado at Boulder. Ph.D.: 123.

PEER REPORTS

PEER reports are available individually or by yearly subscription. PEER reports can be ordered at http://peer.berkeley.edu/publications/peer_reports.html or by contacting the Pacific Earthquake Engineering Research Center, 1301 South 46th Street, Richmond, CA 94804-4698. Tel.: (510) 665-3448; Fax: (510) 665-3456; Email: peer_editor@berkeley.edu

PEER 2009/02 *Improving Earthquake Mitigation through Innovations and Applications in Seismic Science, Engineering, Communication, and Response. Proceedings of a U.S.-Iran Seismic Workshop.* October 2009.

PEER 2009/01 *Evaluation of Ground Motion Selection and Modification Methods: Predicting Median Interstory Drift Response of Buildings.* Curt B. Haselton, Ed. June 2009.

PEER 2008/10 *Technical Manual for Strata.* Albert R. Kottke and Ellen M. Rathje. February 2009.

PEER 2008/09 *NGA Model for Average Horizontal Component of Peak Ground Motion and Response Spectra.* Brian S.-J. Chiou and Robert R. Youngs. November 2008.

PEER 2008/08 *Toward Earthquake-Resistant Design of Concentrically Braced Steel Structures.* Patxi Uriz and Stephen A. Mahin. November 2008.

PEER 2008/07 *Using OpenSees for Performance-Based Evaluation of Bridges on Liquefiable Soils.* Stephen L. Kramer, Pedro Arduino, and HyungSuk Shin. November 2008.

PEER 2008/06 *Shaking Table Tests and Numerical Investigation of Self-Centering Reinforced Concrete Bridge Columns.* Hyung IL Jeong, Junichi Sakai, and Stephen A. Mahin. September 2008.

PEER 2008/05 *Performance-Based Earthquake Engineering Design Evaluation Procedure for Bridge Foundations Undergoing Liquefaction-Induced Lateral Ground Displacement.* Christian A. Ledezma and Jonathan D. Bray. August 2008.

PEER 2008/04 *Benchmarking of Nonlinear Geotechnical Ground Response Analysis Procedures.* Jonathan P. Stewart, Annie On-Lei Kwok, Yousseff M. A. Hashash, Neven Matasovic, Robert Pyke, Zhiliang Wang, and Zhaohui Yang. August 2008.

PEER 2008/03 *Guidelines for Nonlinear Analysis of Bridge Structures in California.* Ady Aviram, Kevin R. Mackie, and Božidar Stojadinović. August 2008.

PEER 2008/02 *Treatment of Uncertainties in Seismic-Risk Analysis of Transportation Systems.* Evangelos Stergiou and Anne S. Kiremidjian. July 2008.

PEER 2008/01 *Seismic Performance Objectives for Tall Buildings.* William T. Holmes, Charles Kircher, William Petak, and Nabih Youssef. August 2008.

PEER 2007/12 *An Assessment to Benchmark the Seismic Performance of a Code-Conforming Reinforced Concrete Moment-Frame Building.* Curt Haselton, Christine A. Goulet, Judith Mitrani-Reiser, James L. Beck, Gregory G. Deierlein, Keith A. Porter, Jonathan P. Stewart, and Ertugrul Taciroglu. August 2008.

PEER 2007/11 *Bar Buckling in Reinforced Concrete Bridge Columns.* Wayne A. Brown, Dawn E. Lehman, and John F. Stanton. February 2008.

PEER 2007/10 *Computational Modeling of Progressive Collapse in Reinforced Concrete Frame Structures.* Mohamed M. Talaat and Khalid M. Mosalam. May 2008.

PEER 2007/09 *Integrated Probabilistic Performance-Based Evaluation of Benchmark Reinforced Concrete Bridges.* Kevin R. Mackie, John-Michael Wong, and Božidar Stojadinović. January 2008.

PEER 2007/08 *Assessing Seismic Collapse Safety of Modern Reinforced Concrete Moment-Frame Buildings.* Curt B. Haselton and Gregory G. Deierlein. February 2008.

PEER 2007/07 *Performance Modeling Strategies for Modern Reinforced Concrete Bridge Columns.* Michael P. Berry and Marc O. Eberhard. April 2008.

PEER 2007/06 *Development of Improved Procedures for Seismic Design of Buried and Partially Buried Structures.* Linda Al Atik and Nicholas Sitar. June 2007.

PEER 2007/05 *Uncertainty and Correlation in Seismic Risk Assessment of Transportation Systems.* Renee G. Lee and Anne S. Kiremidjian. July 2007.

PEER 2007/04 *Numerical Models for Analysis and Performance-Based Design of Shallow Foundations Subjected to Seismic Loading.* Sivapalan Gajan, Tara C. Hutchinson, Bruce L. Kutter, Prishati Raychowdhury, José A. Ugalde, and Jonathan P. Stewart. May 2008.

PEER 2007/03 *Beam-Column Element Model Calibrated for Predicting Flexural Response Leading to Global Collapse of RC Frame Buildings.* Curt B. Haselton, Abbie B. Liel, Sarah Taylor Lange, and Gregory G. Deierlein. May 2008.

PEER 2007/02 *Campbell-Bozorgnia NGA Ground Motion Relations for the Geometric Mean Horizontal Component of Peak and Spectral Ground Motion Parameters.* Kenneth W. Campbell and Yousef Bozorgnia. May 2007.

PEER 2007/01 *Boore-Atkinson NGA Ground Motion Relations for the Geometric Mean Horizontal Component of Peak and Spectral Ground Motion Parameters.* David M. Boore and Gail M. Atkinson. May. May 2007.

PEER 2006/12 *Societal Implications of Performance-Based Earthquake Engineering*. Peter J. May. May 2007.

PEER 2006/11 *Probabilistic Seismic Demand Analysis Using Advanced Ground Motion Intensity Measures, Attenuation Relationships, and Near-Fault Effects.* Polsak Tothong and C. Allin Cornell. March 2007.

PEER 2006/10 *Application of the PEER PBEE Methodology to the I-880 Viaduct.* Sashi Kunnath. February 2007.

PEER 2006/09 *Quantifying Economic Losses from Travel Forgone Following a Large Metropolitan Earthquake.* James Moore, Sungbin Cho, Yue Yue Fan, and Stuart Werner. November 2006.

PEER 2006/08 *Vector-Valued Ground Motion Intensity Measures for Probabilistic Seismic Demand Analysis*. Jack W. Baker and C. Allin Cornell. October 2006.

PEER 2006/07 *Analytical Modeling of Reinforced Concrete Walls for Predicting Flexural and Coupled–Shear-Flexural Responses*. Kutay Orakcal, Leonardo M. Massone, and John W. Wallace. October 2006.

PEER 2006/06 *Nonlinear Analysis of a Soil-Drilled Pier System under Static and Dynamic Axial Loading.* Gang Wang and Nicholas Sitar. November 2006.

PEER 2006/05 *Advanced Seismic Assessment Guidelines.* Paolo Bazzurro, C. Allin Cornell, Charles Menun, Maziar Motahari, and Nicolas Luco. September 2006.

PEER 2006/04 *Probabilistic Seismic Evaluation of Reinforced Concrete Structural Components and Systems.* Tae Hyung Lee and Khalid M. Mosalam. August 2006.

PEER 2006/03 *Performance of Lifelines Subjected to Lateral Spreading*. Scott A. Ashford and Teerawut Juirnarongrit. July 2006.

PEER 2006/02 *Pacific Earthquake Engineering Research Center Highway Demonstration Project.* Anne Kiremidjian, James Moore, Yue Yue Fan, Nesrin Basoz, Ozgur Yazali, and Meredith Williams. April 2006.

PEER 2006/01 *Bracing Berkeley. A Guide to Seismic Safety on the UC Berkeley Campus.* Mary C. Comerio, Stephen Tobriner, and Ariane Fehrenkamp. January 2006.

PEER 2005/16 *Seismic Response and Reliability of Electrical Substation Equipment and Systems.* Junho Song, Armen Der Kiureghian, and Jerome L. Sackman. April 2006.

PEER 2005/15 *CPT-Based Probabilistic Assessment of Seismic Soil Liquefaction Initiation.* R. E. S. Moss, R. B. Seed, R. E. Kayen, J. P. Stewart, and A. Der Kiureghian. April 2006.

PEER 2005/14 *Workshop on Modeling of Nonlinear Cyclic Load-Deformation Behavior of Shallow Foundations*. Bruce L. Kutter, Geoffrey Martin, Tara Hutchinson, Chad Harden, Sivapalan Gajan, and Justin Phalen. March 2006.

PEER 2005/13 *Stochastic Characterization and Decision Bases under Time-Dependent Aftershock Risk in Performance-Based Earthquake Engineering*. Gee Liek Yeo and C. Allin Cornell. July 2005.

PEER 2005/12 *PEER Testbed Study on a Laboratory Building: Exercising Seismic Performance Assessment*. Mary C. Comerio, editor. November 2005.

PEER 2005/11 *Van Nuys Hotel Building Testbed Report: Exercising Seismic Performance Assessment.* Helmut Krawinkler, editor. October 2005.

PEER 2005/10 *First NEES/E-Defense Workshop on Collapse Simulation of Reinforced Concrete Building Structures.* September 2005.

PEER 2005/09 *Test Applications of Advanced Seismic Assessment Guidelines.* Joe Maffei, Karl Telleen, Danya Mohr, William Holmes, and Yuki Nakayama. August 2006.

PEER 2005/08 *Damage Accumulation in Lightly Confined Reinforced Concrete Bridge Columns.* R. Tyler Ranf, Jared M. Nelson, Zach Price, Marc O. Eberhard, and John F. Stanton. April 2006.

PEER 2005/07 *Experimental and Analytical Studies on the Seismic Response of Freestanding and Anchored Laboratory Equipment.* Dimitrios Konstantinidis and Nicos Makris. January 2005.

PEER 2005/06 *Global Collapse of Frame Structures under Seismic Excitations*. Luis F. Ibarra and Helmut Krawinkler. September 2005.

PEER 2005//05 *Performance Characterization of Bench- and Shelf-Mounted Equipment.* Samit Ray Chaudhuri and Tara C. Hutchinson. May 2006.

PEER 2005/04 *Numerical Modeling of the Nonlinear Cyclic Response of Shallow Foundations.* Chad Harden, Tara Hutchinson, Geoffrey R. Martin, and Bruce L. Kutter. August 2005.

PEER 2005/03 *A Taxonomy of Building Components for Performance-Based Earthquake Engineering.* Keith A. Porter. September 2005.

PEER 2005/02 *Fragility Basis for California Highway Overpass Bridge Seismic Decision Making*. Kevin R. Mackie and Božidar Stojadinović. June 2005.

PEER 2005/01 *Empirical Characterization of Site Conditions on Strong Ground Motion.* Jonathan P. Stewart, Yoojoong Choi, and Robert W. Graves. June 2005.

PEER 2004/09 *Electrical Substation Equipment Interaction: Experimental Rigid Conductor Studies.* Christopher Stearns and André Filiatrault. February 2005.

PEER 2004/08 *Seismic Qualification and Fragility Testing of Line Break 550-kV Disconnect Switches.* Shakhzod M. Takhirov, Gregory L. Fenves, and Eric Fujisaki. January 2005.

PEER 2004/07 *Ground Motions for Earthquake Simulator Qualification of Electrical Substation Equipment.* Shakhzod M. Takhirov, Gregory L. Fenves, Eric Fujisaki, and Don Clyde. January 2005.

PEER 2004/06 *Performance-Based Regulation and Regulatory Regimes.* Peter J. May and Chris Koski. September 2004.

PEER 2004/05 *Performance-Based Seismic Design Concepts and Implementation: Proceedings of an International Workshop.* Peter Fajfar and Helmut Krawinkler, editors. September 2004.

PEER 2004/04 *Seismic Performance of an Instrumented Tilt-up Wall Building.* James C. Anderson and Vitelmo V. Bertero. July 2004.

PEER 2004/03 *Evaluation and Application of Concrete Tilt-up Assessment Methodologies.* Timothy Graf and James O. Malley. October 2004.

PEER 2004/02 *Analytical Investigations of New Methods for Reducing Residual Displacements of Reinforced Concrete Bridge Columns.* Junichi Sakai and Stephen A. Mahin. August 2004.

PEER 2004/01 *Seismic Performance of Masonry Buildings and Design Implications.* Kerri Anne Taeko Tokoro, James C. Anderson, and Vitelmo V. Bertero. February 2004.

PEER 2003/18 *Performance Models for Flexural Damage in Reinforced Concrete Columns.* Michael Berry and Marc Eberhard. August 2003.

PEER 2003/17 *Predicting Earthquake Damage in Older Reinforced Concrete Beam-Column Joints.* Catherine Pagni and Laura Lowes. October 2004.

PEER 2003/16 *Seismic Demands for Performance-Based Design of Bridges.* Kevin Mackie and Božidar Stojadinović. August 2003.

PEER 2003/15 *Seismic Demands for Nondeteriorating Frame Structures and Their Dependence on Ground Motions.* Ricardo Antonio Medina and Helmut Krawinkler. May 2004.

PEER 2003/14 *Finite Element Reliability and Sensitivity Methods for Performance-Based Earthquake Engineering.* Terje Haukaas and Armen Der Kiureghian. April 2004.

PEER 2003/13 *Effects of Connection Hysteretic Degradation on the Seismic Behavior of Steel Moment-Resisting Frames.* Janise E. Rodgers and Stephen A. Mahin. March 2004.

PEER 2003/12 *Implementation Manual for the Seismic Protection of Laboratory Contents: Format and Case Studies.* William T. Holmes and Mary C. Comerio. October 2003.

PEER 2003/11 *Fifth U.S.-Japan Workshop on Performance-Based Earthquake Engineering Methodology for Reinforced Concrete Building Structures.* February 2004.

PEER 2003/10 *A Beam-Column Joint Model for Simulating the Earthquake Response of Reinforced Concrete Frames.* Laura N. Lowes, Nilanjan Mitra, and Arash Altoontash. February 2004.

PEER 2003/09 *Sequencing Repairs after an Earthquake: An Economic Approach.* Marco Casari and Simon J. Wilkie. April 2004.

PEER 2003/08 *A Technical Framework for Probability-Based Demand and Capacity Factor Design (DCFD) Seismic Formats*. Fatemeh Jalayer and C. Allin Cornell. November 2003.

PEER 2003/07 *Uncertainty Specification and Propagation for Loss Estimation Using FOSM Methods.* Jack W. Baker and C. Allin Cornell. September 2003.

PEER 2003/06 *Performance of Circular Reinforced Concrete Bridge Columns under Bidirectional Earthquake Loading*. Mahmoud M. Hachem, Stephen A. Mahin, and Jack P. Moehle. February 2003.

PEER 2003/05 *Response Assessment for Building-Specific Loss Estimation.* Eduardo Miranda and Shahram Taghavi. September 2003.

PEER 2003/04 *Experimental Assessment of Columns with Short Lap Splices Subjected to Cyclic Loads.* Murat Melek, John W. Wallace, and Joel Conte. April 2003.

PEER 2003/03 *Probabilistic Response Assessment for Building-Specific Loss Estimation.* Eduardo Miranda and Hesameddin Aslani. September 2003.

PEER 2003/02 *Software Framework for Collaborative Development of Nonlinear Dynamic Analysis Program.* Jun Peng and Kincho H. Law. September 2003.

PEER 2003/01 *Shake Table Tests and Analytical Studies on the Gravity Load Collapse of Reinforced Concrete Frames.* Kenneth John Elwood and Jack P. Moehle. November 2003.

PEER 2002/24 *Performance of Beam to Column Bridge Joints Subjected to a Large Velocity Pulse.* Natalie Gibson, André Filiatrault, and Scott A. Ashford. April 2002.

PEER 2002/23 *Effects of Large Velocity Pulses on Reinforced Concrete Bridge Columns.* Greg L. Orozco and Scott A. Ashford. April 2002.

PEER 2002/22 *Characterization of Large Velocity Pulses for Laboratory Testing.* Kenneth E. Cox and Scott A. Ashford. April 2002.

PEER 2002/21 *Fourth U.S.-Japan Workshop on Performance-Based Earthquake Engineering Methodology for Reinforced Concrete Building Structures.* December 2002.

PEER 2002/20 *Barriers to Adoption and Implementation of PBEE Innovations.* Peter J. May. August 2002.

PEER 2002/19 *Economic-Engineered Integrated Models for Earthquakes: Socioeconomic Impacts.* Peter Gordon, James E. Moore II, and Harry W. Richardson. July 2002.

PEER 2002/18 *Assessment of Reinforced Concrete Building Exterior Joints with Substandard Details.* Chris P. Pantelides, Jon Hansen, Justin Nadauld, and Lawrence D. Reaveley. May 2002.

PEER 2002/17 *Structural Characterization and Seismic Response Analysis of a Highway Overcrossing Equipped with Elastomeric Bearings and Fluid Dampers: A Case Study.* Nicos Makris and Jian Zhang. November 2002.

PEER 2002/16 *Estimation of Uncertainty in Geotechnical Properties for Performance-Based Earthquake Engineering.* Allen L. Jones, Steven L. Kramer, and Pedro Arduino. December 2002.

PEER 2002/15 *Seismic Behavior of Bridge Columns Subjected to Various Loading Patterns.* Asadollah Esmaeily-Gh. and Yan Xiao. December 2002.

PEER 2002/14 *Inelastic Seismic Response of Extended Pile Shaft Supported Bridge Structures.* T.C. Hutchinson, R.W. Boulanger, Y.H. Chai, and I.M. Idriss. December 2002.

PEER 2002/13 *Probabilistic Models and Fragility Estimates for Bridge Components and Systems.* Paolo Gardoni, Armen Der Kiureghian, and Khalid M. Mosalam. June 2002.

PEER 2002/12 *Effects of Fault Dip and Slip Rake on Near-Source Ground Motions: Why Chi-Chi Was a Relatively Mild M7.6 Earthquake.* Brad T. Aagaard, John F. Hall, and Thomas H. Heaton. December 2002.

PEER 2002/11 *Analytical and Experimental Study of Fiber-Reinforced Strip Isolators.* James M. Kelly and Shakhzod M. Takhirov. September 2002.

PEER 2002/10 *Centrifuge Modeling of Settlement and Lateral Spreading with Comparisons to Numerical Analyses***.** Sivapalan Gajan and Bruce L. Kutter. January 2003.

PEER 2002/09 *Documentation and Analysis of Field Case Histories of Seismic Compression during the 1994 Northridge, California, Earthquake.* Jonathan P. Stewart, Patrick M. Smith, Daniel H. Whang, and Jonathan D. Bray. October 2002.

PEER 2002/08 *Component Testing, Stability Analysis and Characterization of Buckling-Restrained Unbonded Braces™.* Cameron Black, Nicos Makris, and Ian Aiken. September 2002.

PEER 2002/07 *Seismic Performance of Pile-Wharf Connections.* Charles W. Roeder, Robert Graff, Jennifer Soderstrom, and Jun Han Yoo. December 2001.

PEER 2002/06 *The Use of Benefit-Cost Analysis for Evaluation of Performance-Based Earthquake Engineering Decisions.* Richard O. Zerbe and Anthony Falit-Baiamonte. September 2001.

PEER 2002/05 *Guidelines, Specifications, and Seismic Performance Characterization of Nonstructural Building Components and Equipment.* André Filiatrault, Constantin Christopoulos, and Christopher Stearns. September 2001.

PEER 2002/04 *Consortium of Organizations for Strong-Motion Observation Systems and the Pacific Earthquake Engineering Research Center Lifelines Program: Invited Workshop on Archiving and Web Dissemination of Geotechnical Data, 4–5 October 2001*. September 2002.

PEER 2002/03 *Investigation of Sensitivity of Building Loss Estimates to Major Uncertain Variables for the Van Nuys Testbed.* Keith A. Porter, James L. Beck, and Rustem V. Shaikhutdinov. August 2002.

PEER 2002/02 *The Third U.S.-Japan Workshop on Performance-Based Earthquake Engineering Methodology for Reinforced Concrete Building Structures*. July 2002.

PEER 2002/01 *Nonstructural Loss Estimation: The UC Berkeley Case Study.* Mary C. Comerio and John C. Stallmeyer. December 2001.

PEER 2001/16 *Statistics of SDF-System Estimate of Roof Displacement for Pushover Analysis of Buildings.* Anil K. Chopra, Rakesh K. Goel, and Chatpan Chintanapakdee. December 2001.

PEER 2001/15 *Damage to Bridges during the 2001 Nisqually Earthquake.* R. Tyler Ranf, Marc O. Eberhard, and Michael P. Berry. November 2001.

PEER 2001/14 *Rocking Response of Equipment Anchored to a Base Foundation.* Nicos Makris and Cameron J. Black. September 2001.

PEER 2001/13 *Modeling Soil Liquefaction Hazards for Performance-Based Earthquake Engineering.* Steven L. Kramer and Ahmed-W. Elgamal. February 2001.

PEER 2001/12 *Development of Geotechnical Capabilities in OpenSees.* Boris Jeremi . September 2001.

PEER 2001/11 *Analytical and Experimental Study of Fiber-Reinforced Elastomeric Isolators*. James M. Kelly and Shakhzod M. Takhirov. September 2001.

PEER 2001/10 *Amplification Factors for Spectral Acceleration in Active Regions.* Jonathan P. Stewart, Andrew H. Liu, Yoojoong Choi, and Mehmet B. Baturay. December 2001.

PEER 2001/09 *Ground Motion Evaluation Procedures for Performance-Based Design.* Jonathan P. Stewart, Shyh-Jeng Chiou, Jonathan D. Bray, Robert W. Graves, Paul G. Somerville, and Norman A. Abrahamson. September 2001.

PEER 2001/08 *Experimental and Computational Evaluation of Reinforced Concrete Bridge Beam-Column Connections for Seismic Performance*. Clay J. Naito, Jack P. Moehle, and Khalid M. Mosalam. November 2001.

PEER 2001/07 *The Rocking Spectrum and the Shortcomings of Design Guidelines*. Nicos Makris and Dimitrios Konstantinidis. August 2001.

PEER 2001/06 *Development of an Electrical Substation Equipment Performance Database for Evaluation of Equipment Fragilities*. Thalia Agnanos. April 1999.

PEER 2001/05 *Stiffness Analysis of Fiber-Reinforced Elastomeric Isolators.* Hsiang-Chuan Tsai and James M. Kelly. May 2001.

PEER 2001/04 *Organizational and Societal Considerations for Performance-Based Earthquake Engineering.* Peter J. May. April 2001.

PEER 2001/03 *A Modal Pushover Analysis Procedure to Estimate Seismic Demands for Buildings: Theory and Preliminary Evaluation.* Anil K. Chopra and Rakesh K. Goel. January 2001.

PEER 2001/02 *Seismic Response Analysis of Highway Overcrossings Including Soil-Structure Interaction.* Jian Zhang and Nicos Makris. March 2001.

PEER 2001/01 *Experimental Study of Large Seismic Steel Beam-to-Column Connections.* Egor P. Popov and Shakhzod M. Takhirov. November 2000.

PEER 2000/10 *The Second U.S.-Japan Workshop on Performance-Based Earthquake Engineering Methodology for Reinforced Concrete Building Structures.* March 2000.

PEER 2000/09 *Structural Engineering Reconnaissance of the August 17, 1999 Earthquake: Kocaeli (Izmit), Turkey.* Halil Sezen, Kenneth J. Elwood, Andrew S. Whittaker, Khalid Mosalam, John J. Wallace, and John F. Stanton. December 2000.

PEER 2000/08 *Behavior of Reinforced Concrete Bridge Columns Having Varying Aspect Ratios and Varying Lengths of Confinement.* Anthony J. Calderone, Dawn E. Lehman, and Jack P. Moehle. January 2001.

PEER 2000/07 *Cover-Plate and Flange-Plate Reinforced Steel Moment-Resisting Connections*. Taejin Kim, Andrew S. Whittaker, Amir S. Gilani, Vitelmo V. Bertero, and Shakhzod M. Takhirov. September 2000.

PEER 2000/06 *Seismic Evaluation and Analysis of 230-kV Disconnect Switches*. Amir S. J. Gilani, Andrew S. Whittaker, Gregory L. Fenves, Chun-Hao Chen, Henry Ho, and Eric Fujisaki. July 2000.

PEER 2000/05 *Performance-Based Evaluation of Exterior Reinforced Concrete Building Joints for Seismic Excitation.* Chandra Clyde, Chris P. Pantelides, and Lawrence D. Reaveley. July 2000.

PEER 2000/04 *An Evaluation of Seismic Energy Demand: An Attenuation Approach.* Chung-Che Chou and Chia-Ming Uang. July 1999.

PEER 2000/03 *Framing Earthquake Retrofitting Decisions: The Case of Hillside Homes in Los Angeles.* Detlof von Winterfeldt, Nels Roselund, and Alicia Kitsuse. March 2000.

PEER 2000/02 *U.S.-Japan Workshop on the Effects of Near-Field Earthquake Shaking.* Andrew Whittaker, ed. July 2000.

PEER 2000/01 *Further Studies on Seismic Interaction in Interconnected Electrical Substation Equipment.* Armen Der Kiureghian, Kee-Jeung Hong, and Jerome L. Sackman. November 1999.

PEER 1999/14 *Seismic Evaluation and Retrofit of 230-kV Porcelain Transformer Bushings.* Amir S. Gilani, Andrew S. Whittaker, Gregory L. Fenves, and Eric Fujisaki. December 1999.

PEER 1999/13 *Building Vulnerability Studies: Modeling and Evaluation of Tilt-up and Steel Reinforced Concrete Buildings.* John W. Wallace, Jonathan P. Stewart, and Andrew S. Whittaker, editors. December 1999.

PEER 1999/12 *Rehabilitation of Nonductile RC Frame Building Using Encasement Plates and Energy-Dissipating Devices.* Mehrdad Sasani, Vitelmo V. Bertero, James C. Anderson. December 1999.

PEER 1999/11 *Performance Evaluation Database for Concrete Bridge Components and Systems under Simulated Seismic Loads.* Yael D. Hose and Frieder Seible. November 1999.

PEER 1999/10 *U.S.-Japan Workshop on Performance-Based Earthquake Engineering Methodology for Reinforced Concrete Building Structures.* December 1999.

PEER 1999/09 *Performance Improvement of Long Period Building Structures Subjected to Severe Pulse-Type Ground Motions.* James C. Anderson, Vitelmo V. Bertero, and Raul Bertero. October 1999.

PEER 1999/08 *Envelopes for Seismic Response Vectors.* Charles Menun and Armen Der Kiureghian. July 1999.

PEER 1999/07 *Documentation of Strengths and Weaknesses of Current Computer Analysis Methods for Seismic Performance of Reinforced Concrete Members.* William F. Cofer. November 1999.

PEER 1999/06 *Rocking Response and Overturning of Anchored Equipment under Seismic Excitations.* Nicos Makris and Jian Zhang. November 1999.

PEER 1999/05 *Seismic Evaluation of 550 kV Porcelain Transformer Bushings.* Amir S. Gilani, Andrew S. Whittaker, Gregory L. Fenves, and Eric Fujisaki. October 1999.

PEER 1999/04 *Adoption and Enforcement of Earthquake Risk-Reduction Measures.* Peter J. May, Raymond J. Burby, T. Jens Feeley, and Robert Wood.

PEER 1999/03 *Task 3 Characterization of Site Response General Site Categories.* Adrian Rodriguez-Marek, Jonathan D. Bray, and Norman Abrahamson. February 1999.

PEER 1999/02 *Capacity-Demand-Diagram Methods for Estimating Seismic Deformation of Inelastic Structures: SDF Systems.* Anil K. Chopra and Rakesh Goel. April 1999.

PEER 1999/01 *Interaction in Interconnected Electrical Substation Equipment Subjected to Earthquake Ground Motions.* Armen Der Kiureghian, Jerome L. Sackman, and Kee-Jeung Hong. February 1999.

PEER 1998/08 *Behavior and Failure Analysis of a Multiple-Frame Highway Bridge in the 1994 Northridge Earthquake.* Gregory L. Fenves and Michael Ellery. December 1998.

PEER 1998/07 *Empirical Evaluation of Inertial Soil-Structure Interaction Effects.* Jonathan P. Stewart, Raymond B. Seed, and Gregory L. Fenves. November 1998.

PEER 1998/06 *Effect of Damping Mechanisms on the Response of Seismic Isolated Structures.* Nicos Makris and Shih-Po Chang. November 1998.

PEER 1998/05 *Rocking Response and Overturning of Equipment under Horizontal Pulse-Type Motions.* Nicos Makris and Yiannis Roussos. October 1998.

PEER 1998/04 *Pacific Earthquake Engineering Research Invitational Workshop Proceedings, May 14–15, 1998: Defining the Links between Planning, Policy Analysis, Economics and Earthquake Engineering.* Mary Comerio and Peter Gordon. September 1998.

PEER 1998/03 *Repair/Upgrade Procedures for Welded Beam to Column Connections.* James C. Anderson and Xiaojing Duan. May 1998.

PEER 1998/02 *Seismic Evaluation of 196 kV Porcelain Transformer Bushings.* Amir S. Gilani, Juan W. Chavez, Gregory L. Fenves, and Andrew S. Whittaker. May 1998.

PEER 1998/01 *Seismic Performance of Well-Confined Concrete Bridge Columns.* Dawn E. Lehman and Jack P. Moehle. December 2000.

ONLINE REPORTS

The following PEER reports are available by Internet only at http://peer.berkeley.edu/publications/peer_reports.html

PEER 2009/104 *Advanced Implementation of Hybrid Simulation.* Andreas H. Schellenberg, Stephen A. Mahin, Gregory L. Fenves. November 2009.

PEER 2009/103 *Performance Evaluation of Innovative Steel Braced Frames.* T. Y. Yang, Jack P. Moehle, and Božidar Stojadinović. August 2009.

PEER 2009/102 *Reinvestigation of Liquefaction and Nonliquefaction Case Histories from the 1976 Tangshan Earthquake.* Robb Eric Moss, Robert E. Kayen, Liyuan Tong, Songyu Liu, Guojun Cai, and Jiaer Wu. August 2009.

PEER 2009/101 *Report of the First Joint Planning Meeting for the Second Phase of NEES/E-Defense Collaborative Research on Earthquake Engineering.* Stephen A. Mahin et al. July 2009.

PEER 2008/104 *Experimental and Analytical Study of the Seismic Performance of Retaining Structures.* Linda Al Atik and Nicholas Sitar. January 2009.

PEER 2008/103 *Experimental and Computational Evaluation of Current and Innovative In-Span Hinge Details in Reinforced Concrete Box-Girder Bridges. Part 1: Experimental Findings and Pre-Test Analysis.* Matias A. Hube and Khalid M. Mosalam. January 2009.

PEER 2008/102 *Modeling of Unreinforced Masonry Infill Walls Considering In-Plane and Out-of-Plane Interaction.* Stephen Kadysiewski and Khalid M. Mosalam. January 2009.

PEER 2008/101 *Seismic Performance Objectives for Tall Buildings.* William T. Holmes, Charles Kircher, William Petak, and Nabih Youssef. August 2008.

PEER 2007/101 *Generalized Hybrid Simulation Framework for Structural Systems Subjected to Seismic Loading.* Tarek Elkhoraibi and Khalid M. Mosalam. July 2007.

PEER 2007/100 *Seismic Evaluation of Reinforced Concrete Buildings Including Effects of Masonry Infill Walls.* Alidad Hashemi and Khalid M. Mosalam. July 2007.